$130.00 122
#2

MAIL ROOM 40 APR 23 1996 PAT. & TRADEMARK OFC.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

	Silvio Micali	Art Unit:	TO BE ASSIGNED
Serial No.:	TO BE ASSIGNED	Examiner:	TO BE ASSIGNED
Filed:	April 23, 1996	Atty. Docket No.:	U/17957-0015
For:	IMPROVED METHOD FOR CERTIFYING PUBLIC KEYS IN A DIGITAL SIGNATURE SCHEME		

PETITION TO MAKE SPECIAL UNDER 37 C.F.R. §1.102(d)

Honorable Commissioner of
Patents and Trademarks
Washington, DC 20231

Sir:

Applicant hereby petitions the Commissioner to make the above-identified application special under 37 C.F.R. §1.102(d) and MPEP §708.02 (VIII). Enclosed herewith is a check for $130.00 for the petition fee set forth in 37 C.F.R. §1.17(i). Applicant submits that acceleration of the examination of this application is warranted since the requirements of MPEP §708.02 are satisfied as follows:

I. The Application is New.

The above-captioned application is being filed on even date herewith.

300 SL 05/03/96 08636854
1 122 130.00 CK U17957

II. <u>All Claims are Directed to a Single Invention</u>.

The claims are directed to a single invention as set forth in independent claims 1 and 23 of the application.

III. <u>Statement regarding preexamination</u>.

A preexamination search was conducted by the inventor, who is a leading scientist in the cryptology field and has many issued U.S. patents and pending U.S. and foreign patent applications in this field. In addition, the inventor knows or is acquainted with many of the other top researchers and companies in this field and with most of the relevant papers that have been published.

IV. <u>Copies of References</u>.

The preexamination search uncovered the following references:

	U.S. Patent No.	Patentee	Issue Year
1.	5,420,927	Micali	05/30/95

<u>Publications</u>:

2. Gennaro, R. et al. "Robust Threshold DSS Signatures" <u>EuroCrypt 96</u> (9 pages).

3. Harn, L. "Group-oriented *(t,n)* threshold digital signature scheme and digital multisignature" <u>IEE Proc.-Comput. Digit. Tech.</u> Vol. 141, No. 5, 307-313 (Sept. 1994).

4. Kent, S. et al., IAB Privacy Task Force, Request for Comments No. 1114, "Privacy Enhancement for Internet Electronic Mail: Part II -- Certificate-Based Key Management" 1-22 (August 1989).

Reference No. 1, above, is a U.S. patent naming a single inventor, Silvio Micali, the inventor of the present application. The patent issued on May 30, 1995. Reference 1 is directed to the subject matter similar to that of the present application (i.e., shortening certificates), but does not disclose in detail or identically claim the subject matter of the present application. In addition, since Reference 1 issued as a U.S. patent issued less than one year ago and is by the same inventor, then Reference 1 cannot be used in connection with a 102/103 rejection of the present application. Reference 1 could be applied in connection with a possible obviousness-type double patenting rejection. We also note that Reference 1 is the subject of a Request for Reexamination filed by Applicant on April 10, 1996.

References 2 and 3, above, are directed to shared digital signature schemes wherein a secret key is divided into a plurality of portions and each portion is provided to one of a plurality of users. Such schemes are discussed on pages 4 and 5 of the present application. A valid digital signing occurs when a predetermined critical number of users operate on a piece of data using their portion of the secret key. These references differ from the present application in that there is no provision in either Reference 2 or Reference 3 to obtain a compact certificate while maintaining accountability using ordinary signatures (as opposed to group signatures) as recited in the claims of the present application.

Reference 4 was submitted in connection with the Request for Reexamination filed on April 10, 1996 in connection with Reference 1, above. Reference 4 is discussed in detail in the Request for Reexamination. Reference 4 is directed to a two-level certification scheme wherein the issuing authority (in this case RSADSI) issues a certificate in response to the approval of an organizational notary (ON). An ON is an authority of an organization empowered to pass data on to RSADSI for certification. Reference 4 does not show, teach or suggest a mechanism for keeping the ON accountable for the certificates which the ON contributes to certify. In fact, Reference 4 specifically discloses that the final certificate will not include an identification or identifying data of the ON.

Although we believe that we have submitted the correct amount to cover the requested Petition to Make Special, the Commissioner is hereby authorized to credit any overpayment or charge any deficiencies to our Deposit Account No. 06-1448. Two originally executed copies of this form are being submitted for this purpose.

Should there be any questions after reviewing this paper, the Examiner is invited to contact the undersigned at (617) 832-1257.

Respectfully submitted,

[illegible]

Dated: April 23, 1996

Donald W. Muirhead
Reg. No. 33,978

Certificate of Express Mail

I, Donald W. Muirhead, do hereby certify that the foregoing documents are being deposited with the United States Postal Service as Express Mail, postage prepaid, "Post Office to Addressee", in an envelope addressed to the Commissioner for Patents & Trademarks, Box Patent Applications, Washington, D.C. 20231 on this date of April 23, 1996.

[illegible]

Donald W. Muirhead
Express Mail Label: RB656894183US
Date of Deposit: April 23, 1996

To appear in EuroCrypt 96

Robust Threshold DSS Signatures

Rosario Gennaro*, Stanisław Jarecki*, Hugo Krawczyk** and Tal Rabin*

Abstract. We present threshold DSS (Digital Signature Standard) signatures where the power to sign is shared by n players such that for a given parameter $t < n/2$ any subset of $2t + 1$ signers can collaborate to produce a valid DSS signature on any given message, but no subset of t corrupted players can forge a signature (in particular, cannot learn the signature key). In addition, we present a robust threshold DSS scheme that can also tolerate $n/3$ players who refuse to participate in the signature protocol. We can also endure $n/4$ maliciously faulty players that generate incorrect partial signatures at the time of signature computation. This results in a highly secure and resilient DSS signature system applicable to the protection of the secret signature key, the prevention of forgery, and increased system availability.

Our results significantly improve over a recent result by Langford from CRYPTO'95 that presents threshold DSS signatures which can stand much smaller subsets of corrupted players, namely, $t \approx \sqrt{n}$, and do not enjoy the robustness property. As in the case of Langford's result, our schemes require no trusted party. Our techniques apply to other threshold ElGamal-like signatures as well. We prove the security of our schemes solely based on the hardness of forging a regular DSS signature.

1 Introduction

Using a threshold signature scheme, digital signatures can be produced by a group of players rather than by one party. In contrast to the regular signature schemes where the signer is a single entity which holds the secret key, in threshold signature schemes the secret key is shared by a group of n players. In order to produce a valid signature on a given message m, individual players produce their *partial signatures* on that message, and then combine them into a full signature on m. A distributed signature scheme achieves threshold $t < n$, if no coalition of t (or less) players can produce a new valid signature, even after the system has produced many signatures on different messages. A signature resulting from a threshold signature scheme is the same as if it was produced by a single signer possessing the full secret signature key. In particular, the validity of this signature can be verified by anyone who has the corresponding unique public verification key. In other words, the fact that the signature was produced in a distributed fashion is transparent to the recipient of the signature.

* MIT Laboratory for Computer Science, 545 Tech Square, Cambridge, MA 02139, USA. Email: {rosario, stasio, talr}@theory.lcs.mit.edu

** IBM T.J.Watson Research Center, PO Box 704, Yorktown Heights, New York 10598, USA. Email: hugo@watson.ibm.com

Threshold signatures are motivated both by the need that arises in some organizations to have a group of employees agree on a given message (or a document) before signing it, as well as by the need to protect signature keys from the attack of internal and external adversaries. The latter becomes increasingly important with the actual deployment of public key systems in practice. The signing power of some entities, (e.g., a government agency, a bank, a certification authority) inevitably invites attackers to try and "steal" this power. The goal of a threshold signature scheme is twofold: To increase the availability of the signing agency, and at the same time to increase the protection against forgery by making it harder for the adversary to learn the secret signature key. Notice that in particular, the threshold approach rules out the naive solution based on traditional secret sharing, where the secret key is shared in a group but reconstructed by a *single* player each time that a signature is to be produced. Such protocol would contradict the requirement that no t (or less) players can ever produce a new valid signature. In threshold schemes, multiple signatures are produced without an exposure or an explicit reconstruction of the secret key.

Threshold signatures are part of a general approach known as *threshold cryptography* which was introduced by the works of Boyd [Boy86], Desmedt [Des88], and Desmedt and Frankel [DF90]. This approach has received considerable attention in the literature; we refer the reader to [Des94] for a survey of the work in this area. Particular examples of solutions to threshold signatures can be found in [DF92, DDFY94] for the case of RSA signatures, and [Har94, Lan95] for ElGamal-type of signatures.

In this work we present a threshold signature system for DSS, the Digital Signature Standard [fST91]. The importance of providing threshold solutions for signatures schemes used in practice, is that those systems are the ones that will be deployed in the real world and hence they are the ones that require real protection. Threshold DSS signatures schemes were recently studied by Langford [Lan95]. DSS signatures turn out to be less amenable to sharing techniques than RSA or even other ElGamal-type of signatures, e.g., see [Har94]. Langford has overcome some of these difficulties in the case of DSS, exhibiting a solution which requires a group of $n = t^2 - t + 1$ players in order to tolerate up to t players that might refuse to participate in the signature protocol. [1] Thus, for n given servers this solution can resist up to $\sqrt{n}$ corrupted parties.

In general, one would like to have higher thresholds, because they achieve increased security at a given system cost (i.e., a given number of servers). In our work we present threshold DSS signature schemes, where in order to achieve a security threshold t we need $2t + 1$ active signers during signature computation; hence, achieving thresholds of up to $\frac{n-1}{2}$. In addition, we improve on [Lan95] by providing a *robust threshold signature scheme* for DSS which can withstand the participation of dishonest signers during the signature computation operation. Namely, we provide a mechanism that succeeds in constructing a valid signature even if the partial signatures contributed by some of the signers are incorrect. The solution in [Lan95] for DSS does not enjoy this property. In fact, without a "correction" capability as in our solution, or at least a

[1] Langford presents some additional schemes but of more limited applicability: a 2-out-of-n scheme that withstands up to one faulty party, and a general t-out-of-n scheme that uses pre-computed tables of one-time shares and that requires a higher level of trust for the generation of these tables. See [Lan95] for details.

detection mechanism for wrong partial signatures, one may need to try an (exponential in t) number $\binom{n}{2t+1}$ of subsets of signers before finding a subset that generates a valid DSS signature. In our case, we achieve a robust threshold solution to DSS signatures tolerating t faults: that is, t or less corrupted players will not be able to forge signatures, and neither will they be able to prevent the system from computing correct signatures by either refusing to cooperate ($t \leq n/3$ in this case) or by behaving in any arbitrary malicious way (in this case $t \leq n/4$.)[2]

Moreover, our schemes do not require trusting any particular party at any time, including the initial secret key generation. This is an important property achieved by some other ElGamal based threshold signature schemes (including the DSS solution in [Lan95]), but not known for threshold RSA signatures. In the complete version of the paper we will present some additional results, including the application of our techniques to solving threshold signatures for other discrete–log based signatures [ElG85, NR94, HPM94].

Remarkably, our solutions for robust threshold DSS signatures can be *proactivized* using the recent techniques of [HJJ+95] (based on proactive secret sharing of the signature key [HJKY95]). In this way, one can keep the DSS signature key fixed for a long time while its shares can be refreshed periodically. An adversary that tries to break the threshold signature scheme needs then to corrupt t servers *in one single period of time* (which may be as short as one day, one week, etc.), as opposed to having the whole lifetime of the key (e.g., 2 years) to do so.

Technical Overview. The threshold DSS signatures schemes need to deal with two technical difficulties. Combining shares of two secrets, a and b, into shares of the product of these secrets, ab; and producing shares for a secret a given the shares of its reciprocal a^{-1} (computations are over a field Z_q). Langford [Lan95] solves both problems by presenting a multiplicative version of secret sharing that results in polynomials of degree $O(t^2)$; this requires a high number of active signers for signature computation and allows for only a small threshold. In our case, we solve the first problem (sharing of a product of secrets) using a single product of polynomials (with combined degree $2t$ resulting in the need for only $2t+1$ active signers). For the second problem, the sharing of a reciprocal, we introduce a simple and novel solution, which does not incur any additional increase in the number of signers. The solution to this problem is of independent interest and has applications to other threshold ElGamal-like signatures. In addition to these techniques we use many tools from other works, such as verifiable secret sharing (both computational and information-theoretic versions), shared generation/distribution of secrets, re-randomization of secret shares, and more. For achieving the robustness of our schemes we apply error correcting techniques due to Berlekamp and Welch [BW] that achieve a very high rate of error correction, which in our scenario translates into supporting higher thresholds. We prove the security of our schemes assuming the infeasibility of forging a regular DSS signature. That is, our schemes are secure if and only if DSS is unforgeable.

[2] The robustness property has been known for some other shared signature schemes, e.g., Harn's solution [Har94] for threshold AMV-signatures enjoys this property. As for threshold RSA, robust solutions have been only recently found (see [FGY96, GJKR96]).

2 The Digital Signature Standard (DSS)

The Digital Signature Standard (DSS) [fST91] is a signature scheme based on the El-Gamal [ElG85] and Schnorr's [Sch91] signature schemes, which was adopted as the US standard digital signature algorithm. In our description of the DSS protocol we follow the notation introduced in [Lan95], which differs from the original presentation of [fST91] by switching k and k^{-1}. This change will allow a clearer presentation of our threshold DSS signature protocols.

Key Generation. A DSS key is composed of public information p, q, g, a public key y and a secret key x, where:
1. p is a prime number of length l where l is a multiple of 64 and $512 \leq l \leq 1024$.
2. q is a 160-bit prime divisor of $p-1$.
3. g is an element of order q in Z_p^*. The triple (p, q, g) is public.
4. x is the secret key of the signer, a random number $1 \leq x < q$.
5. $y = g^x \bmod p$ is the public verification key.

Signature Algorithm. Let m be a hash of the message to be signed. The signer picks a random number k such that $1 \leq k < q$, calculates $k^{-1} \bmod q$, and sets

$$r = (g^{k^{-1}} \bmod p) \bmod q$$
$$s = k(m + xr) \bmod q$$

The pair (r, s) is a signature of m.

Verification Algorithm. A signature (r, s) of a message m can be publicly verified by checking that $r = (g^{ms^{-1}} y^{rs^{-1}} \bmod p) \bmod q$ where s^{-1} is computed modulo q.

3 Model and Definitions

In this section we introduce our communication model and provide definitions of secure threshold signature schemes.

Communication Model. We assume that our computation model is composed of a set of n *players* $\{P_1, \ldots, P_n\}$ who can be modeled by polynomial-time randomized Turing machines. They are connected by a complete network of private (i.e. untappable) point-to-point channels. In addition, the players have access to a dedicated broadcast channel; by dedicated we mean that if player P_i broadcasts a message, it will be recognized by the other players as coming from P_i. These assumptions (privacy of the communication channels and dedication of the broadcast channel) allow us to focus on a high-level description of the protocols. However, it is worth noting that these abstractions can be substituted with standard cryptographic techniques for privacy, commitment and authentication.

The Adversary. We assume that an adversary, $\mathcal{A}$, can corrupt up to t of the n players in the network. We distinguish between three kinds of (increasingly powerful) adversaries:

- An *Eavesdropping Adversary* learns all the information stored at the corrupted nodes and hears all the broadcasted messages.

- A *Halting Adversary* is an eavesdropping adversary that may *also* cause corrupted players to stop sending messages during the execution of the protocol (e.g., by crashing or disconnecting a machine).
- A *Malicious Adversary* is an eavesdropping adversary that may *also* cause corrupted players to divert from the specified protocol in *any* (possibly malicious) way.

We assume that the computational power of the adversary is adequately modeled by a probabilistic polynomial time Turing machine. (In fact, it suffices for our results to assume that the adversary cannot forge regular DSS signatures, which, in turn, implies the infeasibility of computing discrete logarithms.)

Given a protocol $\mathcal{P}$ the *view* of the adversary, denoted by $\mathcal{VIEW}_{\mathcal{A}}(\mathcal{P})$, is defined as the probability distribution (induced by the random coins of the players) on the knowledge of the adversary, namely, the computational history of all the corrupted players, and the public communications and output of the protocol.

Signature Scheme. A signature scheme $\mathcal{S}$ is a triple of efficient randomized algorithms (Key-Gen, Sig, Ver). Key-Gen is the *key generator* algorithm. It outputs a pair (y, x), such that y is the *public key* and x is the *secret key* of the signature scheme. Sig is the *signing* algorithm: on input a message m and the secret key x, it outputs *sig*, a signature of the message m. Ver is the *verification* algorithm. On input a message m, the public key y, and a string *sig*, it checks whether *sig* is a proper signature of m.

Threshold secret sharing. Given a secret value s we say that the values $(s_1, \ldots, s_n)$ constitute a (t, n)-threshold secret sharing of s if t (or less) of these values reveal no information about s, and if there is an efficient algorithm that outputs s having $t + 1$ of the values s_i as inputs.

Threshold signature schemes. Let $\mathcal{S}$=(Key-Gen, Sig, Ver) be a signature scheme. A (t, n)-threshold signature scheme $\mathcal{TS}$ for $\mathcal{S}$ is a pair of protocols (Thresh-Key-Gen, Thresh-Sig) for the set of players $\{P_1, \ldots, P_n\}$.

Thresh-Key-Gen is a distributed key generation protocol used by the players to jointly generate a pair (y, x) of public/private keys. At the end of the protocol the private output of player P_i is a value x_i such that the values $(x_1, \ldots, x_n)$ form a (t, n)-threshold secret sharing of x. The public output of the protocol contains the public key y. The pairs (y, x) of public/secret key pairs are produced by Thresh-Key-Gen with the same probability distribution as if they were generated by Key-Gen protocol of the regular signature scheme S.

Thresh-Sig is the distributed signature protocol. The private input of P_i is the value x_i. The public inputs consist of a message m and the public key y. The output of the protocol is the value $sig = \mathsf{Sig}(m, x)$. (The verification algorithm is, therefore, the same as in the regular signature scheme $\mathcal{S}$.)

Secure Threshold Signature Schemes. Our definition of security includes both *unforgeability* and *robustness*.

Definition 1. We say that a (t, n)-threshold signature scheme $\mathcal{TS}$ =(Thresh-Key-Gen,Thresh-Sig) is *unforgeable*, if no malicious adversary who corrupts at most t players can produce the signature on any new (i.e., previously unsigned) message m, given the view of the protocol Thresh-Key-Gen and of the protocol Thresh-Sig on input messages $m_1, \ldots, m_k$ which the adversary adaptively chose.

This is the analogous to the notion of existential unforgeability under chosen message attack as defined by Goldwasser, Micali, and Rivest [GMR88]. Following [GMR88] one can also define weaker notions of unforgeability.

In order to prove unforgeability we use the concept of *simulatable adversary view* [GMR89, MR92]. Intuitively, this means that the adversary who sees all the information of the corrupted players and the signature of m, could generate by itself all the other public information produced by the protocol Thresh-Sig. In other words, the run of the protocol provides no useful information to the adversary other than the final signature on m.

Definition 2. A threshold signature scheme $\mathcal{TS}$ =(Thresh-Key-Gen,Thresh-Sig) is *simulatable* if the following properties hold:

1. The protocol Thresh-Key-Gen is simulatable. That is, there exists a simulator SIM_1 that, on input the public key y and the public output generated by an execution of Thresh-Key-Gen, can simulate the view of the adversary on that execution.
2. The protocol Thresh-Sig is simulatable. That is, there exists a simulator SIM_2 that, on input the public input of Thresh-Sig (in particular y and m), t shares $x_{i_1}, \ldots, x_{i_t}$ and the signature s of m, can simulate the view of the adversary on an execution of Thresh-Sig that generates s as an output.

This is actually a stronger property than what we need. Indeed it would be enough for us to say that the executions of the protocols Thresh-Key-Gen and Thresh-Sig give the adversary no advantage in forging signatures for the scheme $\mathcal{S}$. In other words, we could allow the adversary to gain knowledge provided that such knowledge is useless for forging. However our stronger definition subsumes this specific goal and provides a proof of security for any of the "flavors" of signature security as listed in [GMR88]. Indeed one can prove that if the underlying signature scheme $\mathcal{S}$ is unforgeable (in any of the flavors of [GMR88]) and $\mathcal{TS}$ is simulatable then $\mathcal{TS}$ is unforgeable (with the same flavor of $\mathcal{S}$)

Robustness means that the protocol will compute a correct output even in the presence of halting or malicious faults. We will talk about (h, c, n)-*robustness* to indicate that the adversary is allowed to halt up to h players and corrupt maliciously up to c players ($h + c \leq t$ where t is total number of corrupted players).

Definition 3. A threshold signature scheme $\mathcal{TS}$ =(Thresh-Key-Gen,Thresh-Sig) is (h, c, n)-*robust* if even in the presence of an adversary who halts h players and corrupts c players ($h + c \leq t$), both Thresh-Key-Gen and Thresh-Sig complete successfully.

A complete formalization of the definition of secure threshold signature schemes can be found in [Gen96].

4 Existing Tools

Here we briefly recall a few known techniques that we use in our solutions.

Shamir's Secret Sharing. [Sha79]
Given a secret σ, choose at random a polynomial $f(x)$ of degree t, such that $f(0) = \sigma$. Give to player P_i a share $\sigma_i \triangleq f(i) \bmod q$ where q is a prime (We use the interpolation values $i = 1, 2, \ldots, n$ for simplicity; any values in Z_q can be used as well.) We will write $(\sigma_1, \ldots, \sigma_n) \overset{(t,n)}{\longleftrightarrow} \sigma \bmod q$ to denote such a sharing. This protocol generates no public output. It can tolerate t eavesdropping faults if $n \geq t+1$ and, additionally, t halting faults if $n \geq 2t+1$. By using error-correcting techniques (as first suggested in [MS81]) the protocol can also tolerate f malicious faults (among the players, excluding the dealer) if $n \geq t+2f+1$. In the following we will refer to this protocol by Shamir-SS.

Feldman's Verifiable Secret Sharing. [Fel87].
This protocol can tolerate up to $\frac{n-1}{2}$ malicious faults *including the dealer*. Like Shamir's scheme, it generates for each player P_i a share σ_i, such that $(\sigma_1, \ldots, \sigma_n) \overset{(t,n)}{\longleftrightarrow} \sigma \bmod q$. If $f(x) = \sum_j a_j x^j$ then the dealer broadcasts the values $\alpha_j = g^{a_j} \bmod p$. This will allow the players to check that the values σ_i really define a secret by checking that $g^{\sigma_i} = \prod \alpha_j^{i^j}$. It will also allow detection of incorrect shares σ_i' at reconstruction time. Notice that the value of the secret is only computationally secure, e.g., the value $g^{a_0} = g^\sigma \bmod p$ is leaked. In the following we will refer to this protocol by Feldman-VSS.

Unconditionally Secure Verifiable Secret Sharing. [FM88, Ped91b].
In contrast to Feldman's VSS protocol, this protocol provides information theoretic secrecy for the shared secret. This is required by some of our techniques in order to achieve provable security. There are two possible implementation of this primitive. One is by Feldman and Micali [FM88] and is based on a bivariate polynomial sharing. Each player receives a share as in Shamir's case plus some extra information that will allow him to check (by exchanging messages with the other players) that the shares do define a polynomial. This implementation tolerates $\frac{n-1}{3}$ malicious faults. Another possible implementation is the one by Pedersen [Ped91b]. In this implementation the private information of player P_i is the value σ_i such that $(\sigma_1, \ldots, \sigma_n) \overset{(t,n)}{\longleftrightarrow} \sigma \bmod p$. The dealer then commits to each share using an unconditionally secure commitment scheme based on the hardness of discrete log (that is the secrecy of the committed value is unconditional, but it is possible to open the commitment in a different way if one is able to solve discrete log.) The commitment has homomorphic properties that allow the players to check that the shares define a secret as in Feldman's VSS. If one assumes that players are not able to open the commitment in different ways, then at reconstruction time bad shares are detected. The scheme tolerates $\frac{n-1}{2}$ malicious faults. Both implementations can be used in our main protocol. In the following we will refer to this protocol as Uncond-Secure-VSS.

Joint Random Secret Sharing. [Ped91a, Ped91b].
In a Joint Random Secret Sharing scheme the players *collectively* choose shares corresponding to a (t, n)-secret sharing of a random value. At the end of such a protocol each

player P_i has a share σ_i, where $(\sigma_1, \ldots, \sigma_n) \overset{(t,n)}{\longleftrightarrow} \sigma$, and σ is uniformly distributed over the interpolation field. As with a regular (t, n)-secret sharing scheme the value σ is kept secret from every player, and even from any coalition of t players. To realize such a protocol, all players act as dealers of a random local secret that they choose. The final share σ_i $(i = 1, \ldots, n)$ is computed as the sum of the shares dealt to P_i by each player (consequently, the joint secret equals the sum of all dealt secrets). It can be shown that as long as all players correctly share their local secrets and (at least) one of these secrets is chosen randomly then the resultant shares interpolate to a random secret σ. In the cases where active corrupted players, that may deviate from the protocol, are considered, one needs to perform the dealing by each player using a verifiable secret sharing protocol. The basic properties of this protocol, namely, the kind of public information it generates, and its fault tolerance are inherited from the underlying secret sharing scheme. In the following we will refer to these protocols as Joint-Shamir-RSS, Joint-Feldman-RSS or Joint-Uncond-Secure-RSS depending which of the secret sharing schemes is used.

Joint Zero Secret Sharing. [BGW88]
This protocol generates a *collective* sharing of a "secret" whose value is zero. Such a protocol is similar to the above joint random secret sharing protocol but instead of local random secrets each player deals a sharing of the value zero. When verifiability is required each player deals its shares using Feldman-VSS. The correct dealing of the value zero is verified by checking that the free coefficient p_0 of each dealing polynomials is 0 (i.e., by checking that $g^{p_0} = 1$). We will refer to this protocol as Joint-Zero-SS. Notice that by adding such "zero-shares" to existent shares of a secret σ, one obtains a randomization of the shares of σ without changing the secret. This is the way we will typically use the Joint-Zero-SS protocol.

5 DSS Threshold Key-generation without a Trusted Party

An instance (p, q, g) of DSS can be generated using a public procedure (e.g., as specified in [fST91]), or using randomness which is jointly provided by the trustees. To generate a pair of public and private keys in a distributed setting *without a trusted party*, we use a *joint verifiable secret sharing* protocol, following the protocol of Pedersen [Ped91a]. That is the players run an execution of Joint-Feldman-RSS (Section 4). The output of such a protocol is a secret sharing $(x_1, \ldots, x_n) \overset{(t,n)}{\longleftrightarrow} x \bmod q$ of a random value x which in addition reveals $y = g^x \bmod p$. The pair (y, x) is taken to be the public/private key pair.

6 Basic Modules of our Solution

We start by introducing two building blocks which are central to our solution for threshold DSS signatures. The first is an elegant and simple procedure for the shared computation of reciprocals. This procedure is used in our protocols in the following way: the players first produce a joint sharing of a random k, and then compute from these shares a sharing of the reciprocal k^{-1}; the latter in turn are used to compute $r = g^{k^{-1}}$

(it is essential for the security of our application that information on k is not revealed during this process).

Problem 1: Computing reciprocals
Given a secret $k \bmod q$ which is shared among players $P_1, \ldots P_n$, generate a sharing of the value $k^{-1} \bmod q$, without revealing information on k and k^{-1}.

Each player P_i holds a share k_i corresponding to a (t, n) secret sharing of k, namely, $(k_1, \ldots, k_n) \overset{(t,n)}{\longleftrightarrow} k$. The computation of shares for k^{-1} is accomplished as follows.

1. The players jointly generate a (t, n) sharing of a *random* element $a \in Z_q$ using any Joint-RSS protocol (Section 4). We denote the resulting shares by $a_1, a_2, \ldots, a_n$, i.e., $(a_1, \ldots, a_n) \overset{(t,n)}{\longleftrightarrow} a$.
2. The players execute a $(2t, n)$ Joint-Zero-SS protocol (Section 4) after which each player P_i holds a share b_i of the "secret" 0. (The implicit interpolation polynomial is of degree $2t$.)
3. The players reconstruct the value $\mu = ka$ by broadcasting the values $k_i a_i + b_i$, and interpolating the corresponding $2t$-degree polynomial.
4. Each player computes his share u_i of k^{-1} by setting $u_i \triangleq \mu^{-1} a_i \bmod q$.

We refer to the above protocol as the Reciprocal Protocol. The following lemmas can be proven concerning this protocol.

Lemma 4. *It holds that* $(u_1, \ldots, u_n) \overset{(t,n)}{\longleftrightarrow} k^{-1}$.

Intuitively, the value μ revealed in the protocol gives no information on k since μ is the product of k with a random element a. This property is stated in the following lemma.

Lemma 5. *(Informal) There exists a simulator $\mathcal{SIM}$ such that for any adversary $\mathcal{A}$ with access to t shares $k_{i_1}, \ldots, k_{i_t}$ of k, $\mathcal{VIEW}_{\mathcal{A}}$(Reciprocal-Protocol$(k_1, \ldots, k_n)$) is computationally indistinguishable from $\mathcal{SIM}(k_{i_1}, \ldots, k_{i_t})$.*

The proofs of the above lemmas are omitted here as they are implicit in the proofs of our protocols.

Problem 2: Multiplication of two secrets.
Given two secrets u and v, which are both shared among the players, compute the product uv, while maintaining both of the original values secret (aside from the obvious information which is revealed from the result).

Given that u and v are each shared by a polynomial of degree t, each player can locally multiply his shares of u and v, and the result will be a share of uv on a polynomial of degree $2t$. Consequently, the value uv can still be reconstructed from a set of $2t + 1$ correct shares. An additional re-randomization procedure (using the Joint-zero-SS protocol of Section 4) is required to protect the secrecy of the multiplied secret; this randomization is essential because a polynomial of degree $2t$ which is the product of two polynomials of degree t is not a random polynomial, and would expose information about u and v.

We note that this solution to the problem of secret multiplication is a simplified version of the the protocols for the same problem presented in [BGW88, CCD88]. (In contrast to those works, in our case secrets are multiplied only once, thus saving most of the complexity of the solutions in the above works which mainly deal with the problem of repetitive multiplication. Even the simplified version of Rabin [Rab95] for repetitive multiplication involves non-trivial zero-knowledge proofs for verifiability.)

7 DSS-Thresh-Sig-1: Eavesdropping & Halting Adversary

In this section we present our basic protocol for generating a distributed DSS signature.

- It is a secure DSS threshold signature scheme in the presence of an Eavesdropping Adversary (Section 3) when the number of players is $n \geq 2t+1$ where t is the number of faults.
- It is a secure DSS threshold signature scheme in the presence of a Halting Adversary (Section 3) when the number of players is $n \geq 3t+1$ where t is the number of faults.

In other words, this protocol preserves security (secrecy and unforgeability) in the presence of less than a half eavesdropping faults. On other hand, this protocol is robust in the presence of an adversary that in addition to eavesdropping can halt the operation of up to a third of the players by, for example, crashing servers, or disconnecting them from the communication lines.

Outline. Initially every player P_i has a share x_i of the secret key x, shared through a polynomial $F(\cdot)$ of degree t, i.e. $(x_1, \ldots, x_n) \overset{(t,n)}{\longleftrightarrow} x \bmod q$. First the players generate distributively a random k (through a random t-degree polynomial $G(\cdot)$) by running the Joint Random Secret Sharing protocol Joint-Shamir-RSS (Section 4). To compute $r = g^{k^{-1}} \bmod p$ without revealing k, the players use a variation of the Reciprocal Protocol (Section 6) where the value $g^{k^{-1}}$ is reconstructed rather than the value k^{-1}. For the generation of the signature's value s, we note that $s = k(m + xr) \bmod q$ corresponds to the constant term of the multiplication polynomial $G(\cdot)(m + rF(\cdot))$. Since the players have shares of both $G(\cdot)$ and $m + rF(\cdot)$, they can compute s by performing the Multiplication Protocol (Section 6). The full description of protocol DSS-Thresh-Sig-1 is presented in Figure 1. It incorporates the multiplication and reciprocal protocols from Section 6.

Notation. In the description of DSS-Thresh-Sig-1 we use the following notation for two share interpolation operations:

- $v =$ Interpolate$(v_1, \ldots, v_n)$. If $\{v_1, \ldots, v_n\}$ $(n \geq 2t+1)$ is a set of values, such that at most t are *null* and all the remaining ones lie on some t-degree polynomial $F(\cdot)$, then $v \triangleq F(0)$. The polynomial can be computed by standard polynomial interpolation.
- $\beta =$ Exp-Interpolate$(w_1, \ldots, w_n)$. If $\{w_1, \ldots, w_n\}$ $(n \geq 2t+1)$ is a set of values such that at most t are *null* and the remaining ones are of the form $g^{a_i} \bmod p$

where the a_i's lie on some t-degree polynomial $G(\cdot)$, then $\beta \triangleq g^{G(0)}$. This can be computed by $\beta = \prod_{i \in V'} w_i^{\lambda_{i,V'}} = \prod_{i \in V'} (g^{G(i)})^{\lambda_{i,V'}}$, where V' is a $(t+1)$-subset of the correct w_i's and $\lambda_{i,V'}$'s are the corresponding Lagrange interpolation coefficients.

Lemma 6. *DSS-*Thresh-Sig-*1 is a simulatable (in particular unforgeable) threshold DSS signature generation protocol in the presence of up to t eavesdropping faults, where the total number of players is $n \geq 2t+1$.*

Lemma 7. *DSS-*Thresh-Sig-*1 is a $(t, 0, n = 3t+1)$-robust threshold DSS signature generation protocol, namely it tolerates up to t eavesdropping and halting faults if the total number of players is $n \geq 3t+1$.*

The proofs of these lemmas follow the same lines of the proof of Theorem 9 in Section 8. From the above lemmas we derive the following:

Theorem 8. *DSS-*Thresh-Sig-*1 is a secure, i.e. robust and unforgeable, threshold DSS signature in the presence of t eavesdropping (halting) faults if the total number of players is $n \geq 2t+1$ ($n \geq 3t+1$)*

8 Robust Threshold DSS Protocols

In this section we present a robust version of protocol DSS-Thresh-Sig-1 which remains secure even in the presence of a fully *malicious adversary*. The protocol, DSS-Thresh-Sig-2, relies on no assumptions beyond the unforgeability of regular DSS signatures, and can tolerate $\frac{n-1}{4}$malicious faults.

Outline. The protocol is very similar to DSS-Thresh-Sig-1. The only difference is that here we need verifiable sharing of secrets since we assume a Malicious Adversary. The random value k is jointly generated by the players using an *unconditionally* secure VSS (Section 4). This guarantees that absolutely no information is leaked on the values k or k^{-1}. Then the players compute r as in DSS-Thresh-Sig-1, with the only difference that now the random value a is jointly generated using Feldman's VSS protocol. As before s is computed from the appropriate shares. Whenever we reconstruct a secret, in order to detect bad shares contributed by malicious players we perform error-correcting using the Berlekamp and Welch decoder [BW]. As before randomization of polynomials (through the joint zero secret sharing protocols) is added in various places in order to hide possible partial information. The full protocol is exhibited in Figure 2

Notation. In the protocol, we use the following notation:

$$v = \text{EC-Interpolate}(v_1, \ldots, v_n)$$

If $\{v_1, \ldots, v_n\}$ ($n = 4t+1$) is a set of values, such that at least $3t$ of the values lie on some $2t$-degree polynomial $F(\cdot)$, then $v \triangleq F(0)$. The polynomial can be computed by using the Berlekamp-Welch decoder [BW].

An important technical contribution of our paper is the simulation and the proof of the security of this protocol. We prove the following theorem:

DSS Signature Generation – Protocol DSS-Thresh-Sig-1

1. **Generate k**
   The trustees generate a secret random value k, uniformly distributed in Z_q, with a polynomial of degree t, using Joint-Shamir-RSS (Section 4), which creates shares $(k_1, \ldots, k_n) \overset{(t,n)}{\longleftrightarrow} k \bmod q$.

   $\boxed{\text{Secret information of } T_i : \text{share } k_i \text{ of } k}$

2. **Generate random polynomials with constant term 0**
   Execute two instances of Joint-Zero-SS with polynomials of degree $2t$. Denote the shares created in these protocols as $\{b_i\}_{i \in \{1 \ldots n\}}$ and $\{c_i\}_{i \in \{1 \ldots n\}}$.

   $\boxed{\text{Secret information of } T_i : \text{shares } b_i, c_i}$

3. **Compute $r = g^{k^{-1}} \bmod p \bmod q$**
   (a) The trustees generate a random value a, uniformly distributed in Z_q^*, with a polynomial of degree t, using Joint-Shamir-RSS, which creates shares $(a_1, \ldots, a_n) \overset{(t,n)}{\longleftrightarrow} a \bmod q$.

   $\boxed{\text{Secret information of } T_i : \text{share } a_i \text{ of } a}$

   (b) Trustee T_i broadcasts $v_i = k_i a_i + b_i \bmod q$ and $w_i = g^{a_i} \bmod p$. If T_i does not participate his values are set to $null$. Notice that $(v_1, \ldots, v_n) \overset{(2t,n)}{\longleftrightarrow} ka \bmod q$.

   $\boxed{\text{Public information: } \{v_i\}_{i \in \{1 \ldots n\}}, \{g^{a_i}\}_{i \in \{1 \ldots n\}}}$

   (c) Trustee T_i locally computes
   - $\mu \triangleq \text{Interpolate}(v_1, \ldots, v_n) \bmod q \qquad [= ka \bmod q]$
   - $\beta \triangleq \text{Exp-Interpolate}(w_1, \ldots, w_n) \bmod p \qquad [= g^a \bmod p]$
   - $r \triangleq \beta^{\mu^{-1}} \bmod p \bmod q \qquad [= (g^a)^{\mu^{-1}} = g^{k^{-1}} \bmod p \bmod q]$

   $\boxed{\text{Public information: } r}$

4. **Generate $s = k(m + xr) \bmod q$**
   (a) Trustee T_i broadcasts $s_i = k_i(m + x_i r) + c_i \bmod q$. If T_i does not participate, his value s_i is set to $null$. Notice that $(s_1, \ldots, s_n) \overset{(2t,n)}{\longleftrightarrow} k(m + xr) \bmod q$.

   $\boxed{\text{Public information: } \{s_i\}_{i \in \{1 \ldots n\}}}$

   (b) Each trustee computes $s \triangleq \text{Interpolate}(s_1, \ldots, s_n) \bmod q$.

   $\boxed{\text{Public information: } s}$

5. **Output (r, s)** as the signature for m.

Fig. 1. DSS-Thresh-Sig-1 – Halting ($n \geq 3t + 1$) or Eavesdropping ($n \geq 2t + 1$) Adversary

3. (a) In the protocol $\mathcal{A}$ receives t shares $a_1, ..., a_t$ of a proper sharing including g^a and g^{a_i} for $1 \leq i \leq n$. As before a_i for $1 \leq i \leq t$ is uniformly distributed in $[0..q-1]$. The values $\hat{a}_i$ for $1 \leq i \leq t$ were choosen by $\mathcal{SIM}$, under the exact same distribution (Step 4), hence the two distributions are the same. The value $g^{\hat{a}}$ was generated by choosing a random value $\hat{\mu}$ uniformly distributed in $[1..q-1]$ and computing $r^{\hat{\mu}}$ which is equal to $g^{k^{-1}\hat{\mu}}$. The value $k^{-1}\hat{\mu}$ is uniformly distributed in $[1..q-1]$ hence the distribution of g^a and $g^{\hat{a}}$ are computationally indistiguishable. The rest of the values $g^{\hat{a}_i}$ for $t+1 \leq i \leq n$, are obtained through a deterministic computation from $g^{\hat{a}}$ and $g^{\hat{a}_i}$ for $1 \leq i \leq t$, hence they too are compuationally indistinguishable from g^{a_i} for $1 \leq i \leq t$.
   (b) The public values $v_1, ..., v_n$ interpolate to some random uniformly distributed value in $[1..q-1]$. The shares $\hat{v}_1, ..., \hat{v}_n$ interpolate the value $\hat{\mu}$ which is random and uniformly distributed in $[1..q-1]$. In addition, the share v_i, for $1 \leq i \leq t$, satisfies that $v_i = k_i a_i + b_i$. The share $\hat{v}_i$, for $1 \leq i \leq t$ was generated in this manner ($\mathcal{SIM}$-Computation Step 5).
4. Same argument as above noting that the shares interpolate the secret s, and that they were properly generated in $\mathcal{SIM}$-Computation Step 6

This completes the proof of Lemma 11.

9 Malicious Adversary, $n \geq 3t + 1$

We have also devised a DSS distributed signature generation protocol which is secure in the presence of a Malicious Adversary when $n \geq 3t + 1$ where t is the number of faults. In other words, it is secure against an adversary who can corrupt at most a third of the players and can make them deviate arbitrarily from their prescribed instructions. For lack of space we present only an outline of the protocol. The details will appear in the complete version of the paper.

However, this algorithm is provably secure only under the following assumption: let p be a prime of the form $p = kq + 1$ where q is another large prime and g an element of order q in Z_p^*. Let G be the subgroup generated by g.

Conjecture 1 *Choose u, v at random, uniformly and independently in Z_q. The following probability distributions on $G \times G$, (g^u mod p, g^v mod p) and (g^u mod p, $g^{u^{-1}}$ mod p) are computationally indistinguishable.*

In other words, we assume that for random u, the value g^u reveals no computational information on the value $g^{u^{-1}}$.

Outline. This protocol differs from the DSS-Thresh-Sig-2 by more extensive use of Feldman-type verifiability instead of using unconditioanlly secure VSS and error-correcting codes. This shift allows for achieving robustness in the presence of larger number of malicious faults (a third instead of one fourth). The random value k is distributively generated using Feldman's VSS. Notice that this expose the value g^k which is extra information that the adversary would not receive from a regular DSS signature. However if Conjecture 1 holds we can claim that this knowledge would not help an adversary in forging signatures (indeed if it did, such an adversary could be used

$\mathcal{SIM}$

Input: public key y, message m, signature (r, s), shares $x_1, \ldots, x_t$

$\mathcal{SIM}$–Computation

1. Pick random value $\hat{k}$ uniformly distributed in $[0..q-1]$. Generate sharing using unconditionally secure VSS for $\hat{k}$, denote the output of the sharing by $\hat{k}_1, \ldots, \hat{k}_n$.
2. Execute two instances of Joint-Zero-SS with polynomials of degree $2t$. Set $\hat{b}_i$, $\hat{c}_i$ $g^{\hat{b}_i}$ and $g^{\hat{c}_i}$ for $1 \leq i \leq n$ to the output of these invocations.
3. Choose a random value $\hat{\mu}$ uniformly distributed in $[0..q-1]$. Denote $r^{\hat{\mu}}$ by $g^{\hat{a}}$. (We stress that $g^{\hat{a}}$ is only a notation for $r^{\hat{\mu}}$; the value $\hat{a}$ is never explicitly computed in the simulation).
4. Choose t random values $\hat{a}_1, \ldots, \hat{a}_t$ uniformly distributed in $[0..q-1]$. Compute $g^{\hat{a}_i}$ for $1 \leq i \leq t$. From the values $g^{\hat{a}}$ and $\hat{a}_1, \ldots, \hat{a}_t$, generate $g^{\hat{a}_i}$ for $t+1 \leq i \leq n$. Note that $g^{\hat{a}_j} = g^{\lambda_{j,0}\hat{a} + \sum_{i=1}^{t} \lambda_{j,i}\hat{a}_i} = (g^{\hat{a}})^{\lambda_{j,0}} g^{\sum_{i=1}^{t} \lambda_{j,i}\hat{a}_i}$ for known values $\lambda_{j,i}$.
5. Compute $\hat{v}_i \triangleq \hat{a}_i\hat{k}_i + \hat{b}_i$ for $1 \leq i \leq t$. Compute share $\hat{v}_i$ for $t+1 \leq i \leq 2t$, such that $\hat{v}_i$ for $1 \leq i \leq 2t$ define a polynomial $f(x)$ of degree $2t$, such that $f(0) = \hat{\mu}$. Complete the shares $\hat{v}_i$ for $2t+1 \leq i \leq n$ so that $\hat{v}_i \triangleq f(i)$.
6. Compute $\hat{s}_i \triangleq \hat{k}_i(m + x_i r) + \hat{c}_i$ for $1 \leq i \leq t$. Compute share $\hat{s}_i$ for $t+1 \leq i \leq 2t$, such that $\hat{s}_1, \ldots, \hat{s}_{2t}$ define a polynomial $g(x)$ of degree $2t$, such that $g(0) = s$. Complete the shares $\hat{s}_i$ for $2t+1 \leq i \leq n$ so that $\hat{s}_i \triangleq g(i)$.

$\mathcal{SIM}$–Conversation

Comment: In each of the following steps we describe the information which $\mathcal{SIM}$ gives to $\mathcal{A}$. Each of these steps relates to the same numbered step in protocol DSS-Thresh-Sig-2.

1. shares $\hat{k}_1, \ldots, \hat{k}_t$
2. shares $\hat{b}_i$, $\hat{c}_i$ for $1 \leq i \leq t$
   public values $g^0 = 1$, $g^{\hat{b}_i}$, $1 \leq i \leq n$
   $g^0 = 1$, $g^{\hat{c}_i}$, $1 \leq i \leq n$
3. (a) shares $\hat{a}_1, \ldots, \hat{a}_t$
   public values $g^{\hat{a}}$ and $g^{\hat{a}_i}$ for $1 \leq i \leq n$
   (b) public values $\hat{v}_1, \ldots, \hat{v}_n$
   (c) twiddles his thumbs
4. public values: $\hat{s}_1, \ldots, \hat{s}_n$

Fig. 3. Simulation Protocol for DSS-Thresh-Sig-2

to distinguish between "reciprocals in the exponent" – where k replaces the value u in the conjecture.) A difficulty arises when in the protocol we need to reveal the product of two secrets (i.e., when using the Multiplication Protocol of section 6). In this case, the public information of Feldman's VSS is not enough to detect faulty players who reveal incorrect multiplication shares. In order to overcome this difficulty we require the players to perform Chaum's zero–knowledge proof of equality of discrete-logs [Cha90] (originally designed in the context of undeniable signatures). The basic idea is that if two secrets a and b are shared with Feldman's VSS, then each player has a share $c_i = a_i b_i$ of $c = ab$. However if we want to reconstruct c, we cannot sieve out bad shares as in Feldman, since we do not have the values g^{c_i} but only g^{a_i} and g^{b_i}. So we require each player to publish $g^{a_i b_i}$ and prove using Chaum's proof that $DL_g(g^{a_i}) = DL_{g^{b_i}}(g^{a_i b_i})$. As before, randomization of polynomials is added when needed in order to protect partial information.

10 Efficiency Considerations

As in the case of the generation of regular DSS signatures the most expensive part of our protocols is the computation of r, as it includes all the modular exponentiations and the interactive exchange of messages between players. However (as in the case of regular DSS signatures) such computation can be performed off-line. In this case the signature generation becomes extremely efficient and non-interactive.

References

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-cryptographic Fault-Tolerant Distributed Computations. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 1–10, 1988.

[Boy86] C. Boyd. Digital Multisignatures. In H. Baker and F. Piper, editors, *Cryptography and Coding*, pages 241–246. Claredon Press, 1986.

[BW] E. Berlekamp and L. Welch. Error correction of algebraic block codes. US Patent 4,633,470.

[CCD88] D. Chaum, C. Crepeau, and I. Damgard. Multiparty Unconditionally Secure Protocols. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 11–19, 1988.

[Cha90] D. Chaum. Zero–knowledge undeniable signatures. In *Proc. EUROCRYPT 90*, pages 458–464. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 473.

[DDFY94] Alredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *Proc. 26th ACM Symp. on Theory of Computing*, pages 522–533, Santa Fe, 1994.

[Des88] Yvo Desmedt. Society and group oriented cryptography: A new concept. In Carl Pomerance, editor, *Proc. CRYPTO 87*, pages 120–127. Springer-Verlag, 1988. Lecture Notes in Computer Science No. 293.

[Des94] Yvo G. Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 5(4):449–457, July 1994.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In G. Brassard, editor, *Proc. CRYPTO 89*, pages 307–315. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 435.

[DF92] Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures. In J. Feigenbaum, editor, *Proc. CRYPTO 91*, pages 457–469. Springer-Verlag, 1992. Lecture Notes in Computer Science No. 576.

[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Info. Theory*, IT 31, 1985.

[Fel87] P. Feldman. A Practical Scheme for Non-Interactive Verifiable Secret Sharing. In *Proc. 28th IEEE Symp. on Foundations of Comp. Science*, pages 427–437, 1987.

[FGY96] Y. Frankel, P. Gemmel, and M. Yung. Witness-based cryptographic program checking and robust function sharing. To appear in proceedings of STOC96, 1996.

[FM88] P. Feldman and S. Micali. An Optimal Algorithm for Synchronous Byzantine Agreement. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 148–161, 1988.

[fST91] National Institute for Standards and Technology. Digital Signature Standard (DSS). Technical Report 169, August 30 1991.

[Gen96] Rosario Gennaro. Theory and practice of verifiable secret sharing. Ph.D. thesis, Massachusetts Institute of Technology, to appear, 1996.

[GJKR96] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust and efficient sharing of rsa functions. manuscript, 1996.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2):281–308, April 1988.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM. J. Computing*, 18(1):186–208, February 1989.

[Har94] L. Harn. Group oriented (t,n) digital signature scheme. *IEEE Proc.-Comput.Digit.Tech*, 141(5), Sept 1994.

[HJJ$^+$95] Amir Herzberg, Markus Jakobson, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive proactive public key and signature systems. manuscript, 1995.

[HJKY95] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing, or: How to cope with perpetual leakage. In *Proc. CRYPTO 95*. Springer-Verlag, August 1995. Lecture Notes in Computer Science No. 963.

[HPM94] P. Horster, H. Petersen, and M. Michels. Meta-elgamal signatures schemes. In *2nd ACM Conference on Computer and Communications Security*, pages 96–107, 1994.

[Lan95] S. Langford. Threshold dss signatures without a trusted party. In *Crypto'95*, pages 397–409. Springer-Verlag, 1995. Lecture Notes in Computer Science No. 963.

[MR92] S. Micali and P. Rogaway. Secure computation. In J. Feigenbaum, editor, *Proc. CRYPTO 91*, pages 392–404. Springer-Verlag, 1992. Lecture Notes in Computer Science No. 576.

[MS81] R. McEliece and D. Sarwate. On sharing secrets and reed-solomon codes. *Communications of the ACM*, 24(9):583–584, September 1981.

[NR94] K. Nyberg and R. Rueppel. Message recovery for signature schemes based on the discrete logarithm problem. In *Proc. EUROCRYPT 94*, pages 175–190, 1994.

[Ped91a] T. Pedersen. Distributed provers with applications to undeniable signatures. In *Proc. EUROCRYPT 91*, 1991.

[Ped91b] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proc. CRYPTO 91*, pages 129–140, 1991.

[Rab95] M. Rabin. A Simplification Approach to Distributed Multiparty Computations. personal communication, 1995.

[Sch91] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.

[Sha79] A. Shamir. How to Share a Secret. *Communications of the ACM*, 22:612–613, 1979.

DSS Signature Generation – Protocol DSS-Thresh-Sig-2

1. **Generate k**
   The trustees generate a secret value k, uniformly distributed in Z_q, by running Joint-Uncond-Secure-RSS with a polynomial of degree t. Notice that this generates $(k_1, \ldots, k_n) \overset{(t,n)}{\longleftrightarrow} k \bmod q$.

   $\boxed{\text{Secret information of } T_i : \text{a share } k_i \text{ of } k}$

2. **Generate random polynomials with constant term 0**
   Execute two instances of Joint-Zero-SS with polynomials of degree $2t$ as underlying scheme. Denote the shares created in these protocols as $\{b_i\}_{i\in\{1\ldots n\}}$ and $\{c_i\}_{i\in\{1\ldots n\}}$.

   Secret information of T_i : shares b_i, c_i
   Public information: $g^0 = 1,\ g^{b_i},\ g^0 = 1,\ g^{c_i},\ 1 \le i \le n$

3. **Generate $r = g^{k^{-1}} \bmod p \bmod q$**
   (a) Generate a random value a, uniformly distributed in Z_q^*, with a polynomial of degree t, using Joint-Feldman-RSS.

   Secret information of T_i : a share a_i of a
   Public information: $g^a,\ g^{a_i},\ 1 \le i \le n$

   (b) Trustee T_i broadcasts $v_i = k_i a_i + b_i \bmod q$. If T_i doesn't broadcast a value set v_i to *null*.

   Public information: $v_1, \ldots, v_n$ where for at least $n - t$ values j it holds that $v_j = k_j a_j + b_j \bmod q$

   (c) Trustee T_i computes locally
   - $\mu \triangleq$ EC-Interpolate$(v_1, \ldots, v_n) \bmod q$ $\quad [= ka \bmod q]$
   - $\mu^{-1} \bmod q$ $\quad [= k^{-1}a^{-1} \bmod q]$
   - $r \triangleq (g^a)^{\mu^{-1}} \bmod p \bmod q$ $\quad [= g^{k^{-1}} \bmod p \bmod q]$

   Note: Even though the above computations are local, as they are done on public information we can assume that:

   $\boxed{\text{Public information: } r}$

4. **Generate $s = k(m + xr) \bmod q$**
   Trustee T_i broadcasts $s_i = k_i(m + x_i r) + c_i \bmod q$.

   Public information: $s_1, \ldots, s_n$ where for at least $n - t$ values j it holds that $s_j = k_j(m + x_j r) + c_j \bmod q$

   Set $s \triangleq$ EC-Interpolate$(s_1, \ldots, s_n)$.
5. **Output** the pair (r, s) as the signature for m

Fig. 2. DSS - Distributed signature generation - Malicious Adversary, $n \ge 4t + 1$

Group-oriented (t, n) threshold digital signature scheme and digital multisignature

L. Harn

Indexing terms: Threshold cryptosystem, Digital signature, Multisignature, Signature verification

Abstract: The paper presents group-oriented (t, n) threshold digital signature schemes based on the difficulty of solving the discrete logarithm problem. By employing these schemes, any t out of n users in a group can represent this group to sign the group signature. The size of the group signature and the verification time of the group signature are equivalent to that of an individual digital signature. In other words, the (t, n) threshold signature scheme has the following five properties: (i) any group signature is mutually generated by at least t group members; (ii) the size of the group signature is equivalent to the size of an individual signature; (iii) the signature verification process is simplified because there is only one group public key required; (iv) the group signature can be verified by any outsider; and (v) the group holds the responsibility to the signed message. In addition to the above properties, two of the schemes proposed do not require the assistance of a mutually trusted party. Each member selects its own secret key and the group public key is determined by all group members. Each group member signs a message separately and sends the individual signature to a designated clerk. The clerk validates each individual signature and then combines all individual signatures into a group signature. The (n, n) threshold signature scheme can be easily extended to become a digital multisignature scheme.

1 Introduction

The threshold cryptosystem was first introduced by Desmedt [4] in 1987. In this system, each group, instead of each group member, publishes a single group public key. An outsider can use this single public key to send an encrypt message to this group. The received ciphertext can only be deciphered properly when the number of participating group members is larger than or equal to the threshold value. All up-to-date solutions for the group-oriented threshold cryptosystem can be classified into the following two categories: (i) solutions with the assistance of a mutually trusted party to decide the group secret key and generate individual secrets for all group members [5, 8, 10]; and (ii) solutions without the assistance of a mutually trusted party [13, 16]. As pointed out by Ingemarsson and Simmons [11], in most applications a trusted party in a group does not exist. This situation becomes more common in some commercial and/or international applications. Thus, the solutions without the assistance of a mutually trusted party become very attractive.

The threshold signature scheme is very similar to the threshold cryptosystem. In a threshold signature scheme, the group signature can only be generated when the number of participating group members is larger than or equal to the threshold value. Any outsider can use a group public key to verify this group signature. Boyd [2] proposed the first (n, n) group-oriented signature based on the RSA assumption [18] in 1986. In his scheme, if the number of group members is larger than two, most of the members can only sign the message blindly. Chaum and van Heyst [3] proposed another (n, n) group-oriented signature scheme in Eurocrypt '91. In their scheme, the number of listed group public key is not limited to one. In 1991, Desmedt and Frankel [6] proposed the first (t, n) threshold digital signature scheme based on the RSA assumption. In their scheme, a trusted key authentication center (KAC) is required for determining the group secret key and all members' secret keys. A group-oriented (n, n) undeniable signature scheme [9] was presented in Auscrypt '92. Unlike the normal signature that can be verified by any outsider, the undeniable signature can only be verified with the cooperation of all signers.

The threshold signature scheme can be applied to solve the problem of issuing checks for a corporation. For security reasons, it may be a company's policy that checks be signed by at least t individuals rather than one person. More formally, a (t, n) threshold digital signature scheme is designed to break the group secret key K into n different 'shadows', $K_1, K_2, \ldots, K_n$, so that:

(i) with knowledge of any t 'shadow' ($t < n$), the group signature can be easily produced;

(ii) with knowledge of any $t - 1$ or fewer 'shadows', it is impossible to forge a group signature;

(iii) it is impossible to derive the group secret key from the released group signature and all partial signatures; and

(iv) it is impossible to derive any secret 'shadow' from the released group signature and all partial signatures.

This definition is very similar to the definition of a (t, n) threshold secret sharing scheme. The major difference is that, in the secret sharing scheme, since the secret 'shadows' are exchanged among users and the group secret key is derived after each secret reconstruction process, the group secret key can only be used once if no other encryption scheme has been used; however, in the signature scheme, since the secret 'shadows' and the

Paper 1293E (C3), first received 6th September 1993 and in revised form 8th March 1994

The author is with the Computer Science Telecommunications Program, University of Missouri – Kansas City, Kansas City, MO 64110, USA

$[1, p-1]$ and computes a corresponding public key as

$$y_i = \alpha^{z_i} \bmod p$$

The group public key y is then determined by all members as

$$y = \prod_{i=1}^{n} y_i \bmod p$$

3.2 Group signature generating phase

The scheme allows group members to sign a message simultaneously. This phase can be further divided into two parts.

Part 1: Generating and verifying individual signature. The procedure for generating an individual signature can be described as follows.

(i) Each member u_i randomly selects a number k_i from $[1, p-1]$ and computes

$$r_i = \alpha^{k_i} \bmod p$$

(ii) The result $\{r_i\}$ is broadcasted to all members. Once r_i, $i = 1, 2, \ldots, n$, from all members are available through the broadcast channel, each member computes the value r as

$$r = \prod_{i=1}^{n} r_i \bmod p$$

(iii) Member u_i uses his secret keys z_i and k_i, to sign the message m based on the modified signature scheme and solves the equation

$$s_i = z_i m' - k_i r \bmod p - 1$$

for integer s_i, where $0 \leqslant s_i \leqslant p-2$ and $m' = f(m)$, and transmits $\{m, s_i\}$ to the clerk. Note here that the individual signature, $\{r_i, s_i\}$, is a partial signature of message m.

Once the clerk receives the individual signature $\{r_i, s_i\}$ from u_i, he needs to verify the validity of this signature. To do this the clerk uses u_i's public key y_i to compute

$$y_i^{m'} = r_i^r \alpha^{s_i} \bmod p$$

where $m' = f(m)$. If the equation holds true, the partial signature $\{r_i, s_i\}$ of message m received from u_i has been verified.

Part 2: Generating the group signature. Once all partial group signatures are received and verified by the clerk, the group signature of message m can be generated as $\{r, s\}$, where $s = s_1 + s_2 + \cdots + s_n \bmod p - 1$.

3.3 Group signature verification phase

After receiving the group signature $\{r, s\}$, of the message m, an outsider needs to use the group public key y to verify the validity of the signature. The verification procedure is given as

$$y^{m'} = r^r \alpha^s \bmod p$$

where $m' = f(m)$. If the equation holds true, the group signature $\{r, s\}$ has been verified.

Theorem: If $y^{m'} = r^r \alpha^s \bmod p$, the group signature $\{r, s\}$ has been verified.

Proof: With the knowledge of secret key z_i, user u_i is able to generate its partial signature $\{r_i, s_i\}$ for message m to satisfy

$$y_i^{m'} = r_i^r \alpha^{s_i} \bmod p$$

where $m' = f(m)$. Multiplying the above equation for $i = 1, 2, \ldots, n$ yields

$$\prod_{i=1}^{n} y_i^{m'} = \prod_{i=1}^{n} r_i^r \alpha^{s_i} \bmod p$$

This relation is the same as

$$\prod_{i=1}^{n} (y_i)^{m'} = \left(\prod_{i=1}^{n} r_i^r\right)(\alpha)^{\sum_{i=1}^{n} s_i} \bmod p$$

Since

$$r = \prod_{i=1}^{n} r_i \bmod p$$

$$s = s_1 + s_2 + \cdots + s_n \bmod p - 1$$

and

$$y = \prod_{i=1}^{n} y_i \bmod p$$

then

$$y^{m'} = r^r \alpha^s \bmod p$$

3.4 Security

The security analysis of this signature scheme is very similar to the security analysis of the modified signature scheme just described. Here, some possible attacks are briefly examined.

(i) Instead of satisfying $y_i^{m'} = r_i^{r_i} \alpha^{s_i} \bmod p$ as in the modified signature scheme, the partial signature in the group signature scheme needs to satisfy $y_i^{m'} = r_i^r \alpha^{s_i} \bmod p$. Since r_i and r are public values and contain no secrets, the attacker cannot reveal any secret from this equation.

(ii) With the knowledge of all partial signatures and the group signature, the attacker needs to solve the equation

$$\begin{aligned}(z_1 + z_2 + \cdots + z_n)m' \\ &= (k_1 + k_2 + \cdots + k_n)r \\ &\quad + (s_1 + s_2 + \cdots + s_n) \bmod p - 1\end{aligned}$$

in order to determine the secret keys. It has the same difficulty as in the modified signature scheme.

(iii) An attacker might try to impersonate user u_i by randomly selecting a r_i' and then obtaining

$$r' = \left(\prod_{j=1, j \neq i}^{n} r_j\right) r_i' \bmod p$$

The attacker needs to find a value s_i' to satisfy the equation as $y_i^{m'} = r_i'^{r'} \alpha^{s_i'} \bmod p$. This difficulty is equivalent to solving the discrete logarithm problem. On the other hand, an attacker might first try to randomly select a pair of (r', s_i'), then broadcast a forged r_i', to satisfy

$$r' = \left(\prod_{j=1, j \neq i}^{n} r_j\right) r_i' \bmod p$$

Since $y_i^{m'} \neq r_i'^{r'} \alpha^{s_i'} \bmod p$, this forged partial signature (r_i', s_i') cannot satisfy the signature verification equation. It is therefore concluded that, although one of the partial signatures r_i from each member is not authenticated by other members and the attacker can easily change this value, to place a successful attack is infeasible. On the other hand, if it is necessary, each member can still sign this partial signature and then make the signature of r_i and r_i itself available on the broadcast channel.

(iv) If the clerk is allowed to collect all r_i from the members and to broadcast the productive result r for all from u_i, he needs to verif[illegible] validity of this partial signature. The clerk uses u_i's public keys, x_i and y_i, and partial signature $\{r_i, s_i\}$ to compute

$$y_i^{m' \prod_{j=1, j\neq i}^{t} \frac{-x_j}{x_i - x_j}} = r_i^{r} \alpha^{s_i} \bmod p$$

where $m' = f(m)$. If the equation holds true, the partial signature $\{r_i, s_i\}$ of message m received from u_i is valid.

Part 2: (t, n) Signature generation. Once t partial signatures are received and verified by the clerk, the group signature of message m can be generated as $\{r, s\}$, where $s = s_1 + s_2 + \cdots + s_t \bmod q$.

4.3 (t, n) Threshold signature verification phase

After receiving the group signature $\{r, s\}$ of the message m, an outsider needs to use the group public key y to verify the validity of the signature. The verification procedure is given as

$$y^{m'} = r^{r} \alpha^{s} \bmod p$$

where $m' = f(m)$. If the equation holds true, the group signature $\{r, s\}$ is valid.

Theorem: If $y^{m'} = r^{r}\alpha^{s} \bmod p$, the group signature $\{r, s\}$ has been verified.

Proof: With the knowledge of secret shadow $f(x_i)$, user u_i is able to generate its partial signature $\{r_i, s_i\}$ for message m to satisfy

$$y_i^{m' \prod_{j=1, j\neq i}^{t} \frac{-x_j}{x_i - x_j}} = r_i^{r} \alpha^{s_i}$$

Multiplying the above equation for $i = 1, 2, \ldots, t$ gives

$$\prod_{i=1}^{t} y_i^{m' \prod_{j=1, j\neq i}^{t} \frac{-x_j}{x_i - x_j}} = \prod_{i=1}^{t} r_i^{r} \alpha^{s_i} \bmod p \tag{3}$$

With the knowledge of t pairs of $(x_i, f(x_i))$, the unique $(t-1)$th degree polynomial, $f(x)$, can be determined as

$$f(x) = \sum_{i=1}^{t} f(x_i) \prod_{j=1, j\neq i}^{t} \frac{x - x_j}{x_i - x_j} \bmod q$$

The left-hand side of eqn. 3 can be rewritten as

$$\alpha^{m' \sum_{i=1}^{t} f(x_i) \prod_{j=1, j\neq i}^{t} \frac{-x_j}{x_i - x_j} \bmod q} \bmod p$$
$$= \alpha^{m' f(0)} \bmod p$$
$$= y^{m'} \bmod p$$

Since the group signature $\{r, s\}$ can be expressed as

$$r = \prod_{i=1}^{t} r_i \bmod p \quad \text{and} \quad s = s_1 + s_2 + \cdots + s_t \bmod q$$

The right-hand side of eqn. 3 can be rewritten as

$$\left(\prod_{i=1}^{t} r_i\right)^{r} \alpha^{\sum_{i=1}^{t} s_i \bmod q} \bmod p$$
$$= r^{r} \alpha^{s} \bmod p$$

4.4 Security analysis

Here, several possible attacks are proposed, but none can successfully break the scheme.

(i) Derivation of the group secret key $f(0)$, and the secret shadows $f(x_i)$ for $i = 1, 2, \ldots, n$, from the group public key, $y = \alpha^{f(0)} \bmod p$, and the public keys for members, y_i [illegible]$^{x_i)} \bmod p$, for $i = 1, 2, \ldots, n$, are equivalent to solving the discrete logarithm problems.

(ii) Derivation of the secret shadow $f(x_i)$, from one or multiple partial signature pairs (r_i, s_i) based on the equation

$$s_i = f(x_i) \times m' \times \left(\prod_{j=1, j\neq i}^{t} \frac{-x_j}{x_i - x_j}\right) - k_i \times r \bmod q$$

has the same difficulty as the modified ElGamal signature scheme.

(iii) Derivation of the group secret key, $f(0)$, from one or multiple group signature pairs (r, s), based on the equation

$$s = f(0) \times m' - k \times r \bmod q$$

has the same difficulty as the modified ElGamal signature scheme.

(iv) An attacker might try to impersonate member u_i by randomly selecting an integer $k_i' \in [1, q-1]$ and broadcasting $r_i' = \alpha^{k_i'} \bmod p$. Since the productive value,

$$r' = \left(\prod_{j=1, j\neq i}^{t} r_j\right) r_i' \bmod p$$

is determined by all t members, without knowing the secret shadow $f(x_i)$, the attacker cannot generate a valid partial signature pair (r_i', s_i'), to satisfy the verification equation as

$$y_i^{m' \prod_{j=1, j\neq i}^{t} \frac{-x_j}{x_i - x_j}} = r_i'^{r'} \alpha^{s_i'} \bmod p$$

5 (t, n) Threshold signature scheme without the assistance of a mutually trusted party

This scheme is the combination of the two previous schemes. The group secret and public keys are determined by all group members according to the (n, n) signature scheme above. Since there is no mutually trusted party, each member acts as one KAC to generate and distribute his secret key to other members according to the (t, n) scheme.

There are some public parameters that should be agreed to by all group members:

(i) p, a large prime modulus, where $2^{511} < p < 2^{512}$,
(ii) q, a prime divisor of $p - 1$, where $2^{159} < q < 2^{160}$,
(iii) α, where $\alpha = h^{(p-1)/q} \bmod p$, h is a random integer with $1 \leqslant h \leqslant p - 1$ such that $h^{(p-1)/q} \bmod p > 1$.

5.1 Public keys generating phase

Each member randomly selects integers, z_i and x_i from $[1, p-1]$ and computes a corresponding public key as

$$y_i = \alpha^{z_i} \bmod p$$

$\{x_i, y_i\}$ are the member's public keys and $\{z_i\}$ is the member's secret key. Then, the group public key y is determined by all members as

$$y = \prod_{i=1}^{n} y_i \bmod p$$

Since there is no mutually trusted party, each member acts as one KAC to use the $(t, n-1)$ secret sharing scheme as we described in the previous section to distribute his secret key to the other $n - 1$ members. Assuming u_i with the secret key z_i, u_i randomly selects a $(t-1)$th degree polynomial, $f_i(x)$, with $f_i(0) = z_i \bmod q$ and computes the secret shadow, $f_i(x_j) \bmod q$, and the public key, $y_{i,j} = \alpha^{f_i(x_j)} \bmod p$, for each member u_j.

4 DESMEDT, Y.: 'Society a[illegible]oup oriented cryptography: a new concept', *in* 'Advances in Cryptology'. Proceedings of *Crypto '87*, pp. 120–127, 16–20 August, 1988

5 DESMEDT, Y., and FRANKEL, Y.: 'Threshold cryptosystem', *in* 'Advances in Cryptology'. Proceedings of *Crypto '89*, pp. 307–315, 20–24 August 1989

6 DESMEDT, Y., and FRANKEL, Y.: 'Shared generation of authenticators', *in* 'Advances in Cryptology'. Proceedings of *Crypto '91*, 11–15 August 1991

7 ELGAMAL, T.: 'A public key cryptosystem and a signature scheme based on discrete logarithms', *IEEE Trans.*, 1985, IT-31, pp. 469–472

8 FRANKEL, Y.: 'A practical protocol for large group oriented networks', *in* 'Advances in Cryptology'. Proceedings of *Eurocrypt '89*, April 1989, pp. 56–61

9 HARN, L., and YANG, S.: 'Group-oriented undeniable signature schemes without the assistance of a mutually trusted party', *in* 'Advances in Cryptology'. Proceedings of *Auscrypt '92*, December 1992

10 HWANG, T.: 'Cryptosystem for group oriented cryptography', *in* 'Advances in Cryptology'. Proceedings of *Eurocrypt '90*, April 1990, pp. 352–360

11 INGEMARSSON, I., and SIMMONS, G.L.: 'A protocol to set up shared secret schemes without the assistance of a mutually trusted party', *in* 'Advances in Cryptology'. Proceedings of *Eurocrypt '90*, May 21–24, 1990, pp. 266–282

12 KIESLER, T.[illegible]d HARN, L.: 'RSA blocking and multisignature schemes with no bit expansion', *Electronics Letters*, 1990, 26, (18), pp. 1490–1491

13 LAIH, C.S., and HARN, L.: 'Generalized threshold cryptosystems', *in* 'Advances in Cryptology'. Proceedings of *Asiacrypt '91*, Nov. 11–14, 1991, pp. 159–169

14 OHTA, K., and OKAMOTO, T.: 'A digital multisignature scheme based on the Fiat–Shamir scheme', *in* 'Advances in Cryptology'. Proceedings of *Asiacrypt '91*, Nov. 11–14, 1991, pp. 139–148

15 OKAMOTO, T.: 'A digital multisignature scheme using bijective public-key cryptosystems', *ACM Trans. on Comp. Systems*, 1988, 6, (8), pp. 432–441

16 PEDERSEN, T.P.: 'A threshold cryptosystem without a trusted party', *in* 'Advances in Cryptology'. Proceedings of *Eurocrypt '91*, Apr. 8–11, 1991, pp. 522–526

17 POHLIG, S., and HELLMAN, M.: 'An improved algorithm for computing logarithms over GF(p) and its cryptographic significance', *IEEE Trans.*, 1978, IT-24, 106–110

18 RIVEST, R.L., SHAMIR, A., and ADELMAN, L.: 'A method for obtaining digital signatures and public-key cryptosystem', *Commun. of ACM*, 1978, 21, (2), pp. 120–126

19 SHAMIR, A.: 'How to share a secret', *Comm. ACM*, 1979, 22, pp. 612–613

20 'The digital signature standard', *Comm. ACM*, 1992, 35, (7), pp. 36–40

Network Working Group
Request for Comments: 1114

S. Kent
BBNCC
J. Linn
DEC
IAB Privacy Task Force

August 1989

Privacy Enhancement for Internet Electronic Mail: Part II -- Certificate-Based Key Management

STATUS OF THIS MEMO

This RFC suggests a draft standard elective protocol for the Internet community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

ACKNOWLEDGMENT

This RFC is the outgrowth of a series of IAB Privacy Task Force meetings and of internal working papers distributed for those meetings. We would like to thank the members of the Privacy Task Force for their comments and contributions at the meetings which led to the preparation of this RFC: David Balenson, Curt Barker, Matt Bishop, Morrie Gasser, Russ Housley, Dan Nessett, Mike Padlipsky, Rob Shirey, and Steve Wilbur.

Table of Contents

1. Executive Summary

This is one of a series of RFCs defining privacy enhancement mechanisms for electronic mail transferred using Internet mail protocols. RFC-1113 (the successor to RFC 1040) prescribes protocol extensions and processing procedures for RFC-822 mail messages, given that suitable cryptographic keys are held by originators and recipients as a necessary precondition. RFC-1115 specifies algorithms for use in processing privacy-enhanced messages, as called for in RFC-1113. This RFC defines a supporting key management architecture and infrastructure, based on public-key certificate techniques, to provide keying information to message originators and recipients. A subsequent RFC, the fourth in this series, will provide detailed specifications, paper and electronic application forms, etc. for the key management infrastructure described herein.

The key management architecture described in this RFC is compatible with the authentication framework described in X.509. The major contributions of this RFC lie not in the specification of computer communication protocols or algorithms but rather in procedures and conventions for the key management infrastructure. This RFC incorporates numerous conventions to facilitate near term implementation. Some of these conventions may be superceded in time as the motivations for them no longer apply, e.g., when X.500 or similar directory servers become well established.

The RSA cryptographic algorithm, covered in the U.S. by patents administered through RSA Data Security, Inc. (hereafter abbreviated RSADSI) has been selected for use in this key management system. This algorithm has been selected because it provides all the necessary algorithmic facilities, is "time tested" and is relatively efficient to implement in either software or hardware. It is also the primary algorithm identified (at this time) for use in international standards where an asymmetric encryption algorithm is required. Protocol facilities (e.g., algorithm identifiers) exist to permit use of other asymmetric algorithms if, in the future, it becomes appropriate to employ a different algorithm for key management. However, the infrastructure described herein is specific to use of the RSA algorithm in many respects and thus might be different if the underlying algorithm were to change.

Current plans call for RSADSI to act in concert with subscriber organizations as a "certifying authority" in a fashion described

later in this RFC. RSADSI will offer a service in which it will sign a certificate which has been generated by a user and vouched for either by an organization or by a Notary Public. This service will carry a $25 biennial fee which includes an associated license to use the RSA algorithm in conjunction with privacy protection of electronic mail. Users who do not come under the purview of the RSA patent, e.g., users affiliated with the U.S. government or users outside of the U.S., may make use of different certifying authorities and will not require a license from RSADSI. Procedures for interacting with these other certification authorities, maintenance and distribution of revoked certificate lists from such authorities, etc. are outside the scope of this RFC. However, techniques for validating certificates issued by other authorities are contained within the RFC to ensure interoperability across the resulting jurisdictional boundaries.

2. Overview of Approach

This RFC defines a key management architecture based on the use of public-key certificates, in support of the message encipherment and authentication procedures defined in RFC-1113. In the proposed architecture, a "certification authority" representing an organization applies a digital signature to a collection of data consisting of a user's public component, various information that serves to identify the user, and the identity of the organization whose signature is affixed. (Throughout this RFC we have adopted the terms "private component" and "public component" to refer to the quantities which are, respectively, kept secret and made publically available in asymmetric cryptosystems. This convention is adopted to avoid possible confusion arising from use of the term "secret key" to refer to either the former quantity or to a key in a symmetric cryptosystem.) This establishes a binding between these user credentials, the user's public component and the organization which vouches for this binding. The resulting signed, data item is called a certificate. The organization identified as the certifying authority for the certificate is the "issuer" of that certificate.

In signing the certificate, the certification authority vouches for the user's identification, especially as it relates to the user's affiliation with the organization. The digital signature is affixed on behalf of that organization and is in a form which can be recognized by all members of the privacy-enhanced electronic mail community. Once generated, certificates can be stored in directory servers, transmitted via unsecure message exchanges, or distributed via any other means that make certificates easily accessible to message originators, without regard for the security of the transmission medium.

Prior to sending an encrypted message, an originator must acquire a certificate for each recipient and must validate these certificates. Briefly, validation is performed by checking the digital signature in the certificate, using the public component of the issuer whose private component was used to sign the certificate. The issuer's public component is made available via some out of band means (described later) or is itself distributed in a certificate to which this validation procedure is applied recursively.

Once a certificate for a recipient is validated, the public component contained in the certificate is extracted and used to encrypt the data encryption key (DEK) that is used to encrypt the message itself.

The resulting encrypted DEK is incorporated into the X-Key-Info field of the message header. Upon receipt of an encrypted message, a recipient employs his secret component to decrypt this field, extracting the DEK, and then uses this DEK to decrypt the message.

In order to provide message integrity and data origin authentication, the originator generates a message integrity code (MIC), signs (encrypts) the MIC using the secret component of his public-key pair, and includes the resulting value in the message header in the X-MIC-Info field. The certificate of the originator is also included in the header in the X-Certificate field as described in RFC-1113, in order to facilitate validation in the absence of ubiquitous directory services. Upon receipt of a privacy enhanced message, a recipient validates the originator's certificate, extracts the public component from the certificate, and uses that value to recover (decrypt) the MIC. The recovered MIC is compared against the locally calculated MIC to verify the integrity and data origin authenticity of the message.

3. Architecture

3.1 Scope and Restrictions

The architecture described below is intended to provide a basis for managing public-key cryptosystem values in support of privacy enhanced electronic mail (see RFC-1113) in the Internet environment. The architecture describes procedures for ordering certificates from issuers, for generating and distributing certificates, and for "hot listing" of revoked certificates. Concurrent with the issuance of this RFC, RFC 1040 has been updated and reissued as RFC-1113 to describe the syntax and semantics of new or revised header fields used to transfer certificates, represent the DEK and MIC in this public-key context, and to segregate algorithm definitions into a separate RFC to facilitate the addition of other algorithms in the future. This RFC focuses on the management aspects of certificate-

based, public-key cryptography for privacy enhanced mail while RFC-1113 addresses representation and processing aspects of such mail, including changes required by this key management technology.

The proposed architecture imposes conventions for certification paths which are not strictly required by the X.509 recommendation nor by the technology itself. The decision to impose these conventions is based in part on constraints imposed by the status of the RSA cryptosystem within the U.S. as a patented algorithm, and in part on the need for an organization to assume operational responsibility for certificate management in the current (minimal) directory system infrastructure for electronic mail. Over time, we anticipate that some of these constraints, e.g., directory service availability, will change and the procedures specified in the RFC will be reviewed and modified as appropriate.

At this time, we propose a system in which user certificates represent the leaves in a shallow (usually two tier) certification hierarchy (tree). Organizations which act as issuers are represented by certificates higher in the tree. This convention minimizes the complexity of validating user certificates by limiting the length of "certification paths" and by making very explicit the relationship between a certificate issuer and a user. Note that only organizations may act as issuers in the proposed architecture; a user certificate may not appear in a certification path, except as the terminal node in the path. These conventions result in a certification hierarchy which is a compatible subset of that permitted under X.509, with respect to both syntax and semantics.

The RFC proposes that RSADSI act as a "co-issuer" of certificates on behalf of most organizations. This can be effected in a fashion which is "transparent" so that the organizations appear to be the issuers with regard to certificate formats and validation procedures. This is effected by having RSADSI generate and hold the secret components used to sign certificates on behalf of organizations. The motivation for RSADSI's role in certificate signing is twofold. First, it simplifies accounting controls in support of licensing, ensuring that RSADSI is paid for each certificate. Second, it contributes to the overall integrity of the system by establishing a uniform, high level of protection for the private-components used to sign certificates. If an organization were to sign certificates directly on behalf of its affiliated users, the organization would have to establish very stringent security and accounting mechanisms and enter into (elaborate) legal agreements with RSADSI in order to provide a comparable level of assurance. Requests by organizations to perform direct certificate signing will be considered on a case-by-case basis, but organizations are strongly urged to make use of the facilities proposed by this RFC.

Note that the risks associated with disclosure of an organization's secret component are different from those associated with disclosure of a user's secret component. The former component is used only to sign certificates, never to encrypt message traffic. Thus the exposure of an organization's secret component could result in the generation of forged certificates for users affiliated with that organization, but it would not affect privacy-enhanced messages which are protected using legitimate certificates. Also note that any certificates generated as a result of such a disclosure are readily traceable to the issuing authority which holds this component, e.g., RSADSI, due to the non-repudiation feature of the digital signature. The certificate registration and signing procedures established in this RFC would provide non-repudiable evidence of disclosure of an organization's secret component by RSADSI. Thus this RFC advocates use of RSADSI as a co-issuer for certificates until such time as technical security mechanisms are available to provide a similar, system-wide level of assurance for (distributed) certificate signing by organizations.

We identify two classes of exceptions to this certificate signing paradigm. First, the RSA algorithm is patented only within the U.S., and thus it is very likely that certificate signing by issuers will arise outside of the U.S., independent of RSADSI. Second, the research that led to the RSA algorithm was sponsored by the National Science Foundation, and thus the U.S. government retains royalty-free license rights to the algorithm. Thus the U.S. government may establish a certificate generation facilities for its affiliated users. A number of the procedures described in this document apply only to the use of RSADSI as a certificate co-issuer; all other certificate generation practices lie outside the scope of this RFC.

This RFC specifies procedures by which users order certificates either directly from RSADSI or via a representative in an organization with which the user holds some affiliation (e.g., the user's employer or educational institution). Syntactic provisions are made which allow a recipient to determine, to some granularity, which identifying information contained in the certificate is vouched for by the certificate issuer. In particular, organizations will usually be vouching for the affiliation of a user with that organization and perhaps a user's role within the organization, in addition to the user's name. In other circumstances, as discussed in section 3.3.3, a certificate may indicate that an issuer vouches only for the user's name, implying that any other identifying information contained in the certificate may not have been validated by the issuer. These semantics are beyond the scope of X.509, but are not incompatible with that recommendation.

The key management architecture described in this RFC has been

designed to support privacy enhanced mail as defined in this RFC, RFC-1113, and their successors. Note that this infrastructure also supports X.400 mail security facilities (as per X.411) and thus paves the way for transition to the OSI/CCITT Message Handling System paradigm in the Internet in the future. The certificate issued to a user for the $25 biennial fee will grant to the user identified by that certificate a license from RSADSI to employ the RSA algorithm for certificate validation and for encryption and decryption operations in this electronic mail context. No use of the algorithm outside the scope defined in this RFC is authorized by this license as of this time. Expansion of the license to other Internet security applications is possible but not yet authorized. The license granted by this fee does not authorize the sale of software or hardware incorporating the RSA algorithm; it is an end-user license, not a developer's license.

3.2 Relation to X.509 Architecture

CCITT 1988 Recommendation X.509, "The Directory - Authentication Framework", defines a framework for authentication of entities involved in a distributed directory service. Strong authentication, as defined in X.509, is accomplished with the use of public-key cryptosystems. Unforgeable certificates are generated by certification authorities; these authorities may be organized hierarchically, though such organization is not required by X.509. There is no implied mapping between a certification hierarchy and the naming hierarchy imposed by directory system naming attributes. The public-key certificate approach defined in X.509 has also been adopted in CCITT 1988 X.411 in support of the message handling application.

This RFC interprets the X.509 certificate mechanism to serve the needs of privacy-enhanced mail in the Internet environment. The certification hierarchy proposed in this RFC in support of privacy enhanced mail is intentionally a subset of that allowed under X.509. In large part constraints have been levied in order to simplify certificate validation in the absence of a widely available, user-level directory service. The certification hierarchy proposed here also embodies semantics which are not explicitly addressed by X.509, but which are consistent with X.509 precepts. The additional semantic constraints have been adopted to explicitly address questions of issuer "authority" which we feel are not well defined in X.509.

3.3 Entities' Roles and Responsibilities

One way to explain the architecture proposed by this RFC is to examine the various roles which are defined for various entities in the architecture and to describe what is required of each entity in order for the proposed system to work properly. The following sections identify three different types of entities within this architecture: users and user agents, organizational notaries, and certification authorities. For each class of entity we describe the (electronic and paper) procedures which the entity must execute as part of the architecture and what responsibilities the entity assumes as a function of its role in the architecture. Note that the infrastructure described here applies to the situation wherein RSADSI acts as a co-issuer of certificates, sharing the role of certification authority as described later. Other certifying authority arrangements may employ different procedures and are not addressed by this RFC.

3.3.1 Users and User Agents

The term User Agent (UA) is taken from CCITT X.400 Message Handling Systems (MHS) Recommendations, which define it as follows: "In the context of message handling, the functional object, a component of MHS, by means of which a single direct user engages in message handling." UAs exchange messages by calling on a supporting Message Transfer Service (MTS).

A UA process supporting privacy-enhanced mail processing must protect the private component of its associated entity (ordinarily, a human user) from disclosure. We anticipate that a user will employ ancillary software (not otherwise associated with the UA) to generate his public/private component pair and to compute the (one-way) message hash required by the registration procedure. The public component, along with information that identifies the user, will be transferred to an organizational notary (see below) for inclusion in an order to an issuer. The process of generating public and private components is a local matter, but we anticipate Internet-wide distribution of software suitable for component-pair generation to facilitate the process. The mechanisms used to transfer the public component and the user identification information must preserve the integrity of both quantities and bind the two during this transfer.

This proposal establishes two ways in which a user may order a certificate, i.e., through the user's affiliation with an organization or directly through RSADSI. In either case, a user will be required to send a paper order to RSADSI on a form described in a subsequent RFC and containing the following information:

1. Distinguished Name elements (e.g., full legal name, organization name, etc.)

2. Postal address

3. Internet electronic mail address

4. A message hash function, binding the above information to the user's public component

Note that the user's public component is NOT transmitted via this paper path. In part the rationale here is that the public component consists of many (>100) digits and thus is prone to error if it is copied to and from a piece of paper. Instead, a message hash is computed on the identifying information and the public component and this (smaller) message hash value is transmitted along with the identifying information. Thus the public component is transferred only via an electronic path, as described below.

If the user is not affiliated with an organization which has established its own "electronic notary" capability (an organization notary or "ON" as discussed in the next section), then this paper registration form must be notarized by a Notary Public. If the user is affiliated with an organization which has established one or more ONs, the paper registration form need not carry the endorsement of a Notary Public. Concurrent with the paper registration, the user must send the information outlined above, plus his public component, either to his ON, or directly to RSADSI if no appropriate ON is available to the user. Direct transmission to RSADSI of this information will be via electronic mail, using a representation described in a subsequent RFC. The paper registration must be accompanied by a check or money order for $25 or an organization may establish some other billing arrangement with RSADSI. The maximum (and default) lifetime of a certificate ordered through this process is two years.

The transmission of ID information and public component from a user to his ON is a local matter, but we expect electronic mail will also be the preferred approach in many circumstances and we anticipate general distribution of software to support this process. Note that it is the responsibility of the user and his organization to ensure the integrity of this transfer by some means deemed adequately secure for the local computing and communication environment. There is no requirement for secrecy in conjunction with this information transfer, but the integrity of the information must be ensured.

3.3.2 Organizational Notaries

An organizational notary is an individual who acts as a clearinghouse for certificate orders originating within an administrative domain such as a corporation or a university. An ON represents an organization or organizational unit (in X.500 naming terms), and is assumed to have some independence from the users on whose behalf

certificates are ordered. An ON will be restricted through mechanisms implemented by the issuing authority, e.g., RSADSI, to ordering certificates properly associated with the domain of that ON. For example, an ON for BBN should not be able to order certificates for users affiliated with MIT or MITRE, nor vice versa. Similarly, if a corporation such as BBN were to establish ONs on a per-subsidiary basis (corresponding to organization units in X.500 naming parlance), then an ON for the BBN Communications subsidiary should not be allowed to order a certificate for a user who claims affiliation with the BBN Software Products subsidiary.

It can be assumed that the set of ONs changes relatively slowly and that the number of ONs is relatively small in comparison with the number of users. Thus a more extensive, higher assurance process may reasonably be associated with ON accreditation than with per-user certificate ordering. Restrictions on the range of information which an ON is authorized to certify are established as part of this more elaborate registration process. The procedures by which organizations and organizational units are established in the RSADSI database, and by which ONs are registered, will be described in a subsequent RFC.

An ON is responsible for establishing the correctness and integrity of information incorporated in an order, and will generally vouch for (certify) the accuracy of identity information at a granularity finer than that provided by a Notary Public. We do not believe that it is feasible to enforce uniform standards for the user certification process across all ONs, but we anticipate that organizations will endeavor to maintain high standards in this process in recognition of the "visibility" associated with the identification data contained in certificates. An ON also may constrain the validity period of an ordered certificate, restricting it to less than the default two year interval imposed by the RSADSI license agreement.

An ON participates in the certificate ordering process by accepting and validating identification information from a user and forwarding this information to RSADSI. The ON accepts the electronic ordering information described above (Distinguished Name elements, mailing address, public component, and message hash computed on all of this data) from a user. (The representation for user-to-ON transmission of this data is a local matter, but we anticipate that the encoding specified for ON-to-RSADSI representation of this data will often be employed.) The ON sends an integrity-protected (as described in RFC-1113) electronic message to RSADSI, vouching for the correctness of the binding between the public component and the identification data. Thus, to support this function, each ON will hold a certificate as an individual user within the organization which he represents. RSADSI will maintain a database which identifies the

users who also act as ONs and the database will specify constraints on credentials which each ON is authorized to certify. The electronic mail representation for a user's certificate data in an ON message to RSADSI will be specified in a subsequent RFC.

3.3.3 Certification Authorities

In X.509 the term "certification authority" is defined as "an authority trusted by one or more users to create and assign certificates". This alternate expansion for the acronym "CA" is roughly equivalent to that contemplated as a "central authority" in RFC-1040 and RFC-1113. The only difference is that in X.509 there is no requirement that a CA be a distinguished entity or that a CA serve a large number of users, as envisioned in these RFCs. Rather, any user who holds a certificate can, in the X.509 context, act as a CA for any other user. As noted above, we have chosen to restrict the role of CA in this electronic mail environment to organizational entities, to simplify the certificate validation process, to impose semantics which support organizational affiliation as a basis for certification, and to facilitate license accountability.

In the proposed architecture, individuals who are affiliated with (registered) organizations will go through the process described above, in which they forward their certificate information to their ON for certification. The ON will, based on local procedures, verify the accuracy of the user's credentials and forward this information to RSADSI using privacy-enhanced mail to ensure the integrity and authenticity of the information. RSADSI will carry out the actual certificate generation process on behalf of the organization represented by the ON. Recall that it is the identity of the organization which the ON represents, not the ON's identity, which appears in the issuer field of the user certificate. Therefore it is the private component of the organization, not the ON, which is used to sign the user certificate.

In order to carry out this procedure RSADSI will serve as the repository for the private components associated with certificates representing organizations or organizational units (but not individuals). In effect the role of CA will be shared between the organizational notaries and RSADSI. This shared role will not be visible in the syntax of the certificates issued under this arrangement nor is it apparent from the validation procedure one applies to these certificates. In this sense, the role of RSADSI as the actual signer of certificates on behalf of organizations is transparent to this aspect of system operation.

If an organization were to carry out the certificate signing process locally, and thus hold the private component associated with its

organization certificate, it would need to contact RSADSI to discuss security safeguards, special legal agreements, etc. A number of requirements would be imposed on an organization if such an approach were persued. The organization would be required to execute additional legal instruments with RSADSI, e.g., to ensure proper accounting for certificates generated by the organization. Special software will be required to support the certificate signing process, distinct from the software required for an ON. Stringent procedural, physical, personnel and computer security safeguards would be required to support this process, to maintain a relatively high level of security for the system as a whole. Thus, at this time, it is not recommended that organizations pursue this approach although local certificate generation is not expressly precluded by the proposed architecture.

RSADSI has offered to operate a service in which it serves as a CA for users who are not affiliated with any organization or who are affiliated with an organization which has not opted to establish an organizational notary. To distinguish certificates issued to such "non-affiliated" users the distinguished string "Notary" will appear as the organizational unit name of the issuer of the certificate. This convention will be employed throughout the system. Thus not only RSADSI but any other organization which elects to provide this type of service to non-affiliated users may do so in a standard fashion. Hence a corporation might issue a certificate with the "Notary" designation to students hired for the summer, to differentiate them from full-time employees. At least in the case of RSADSI, the standards for verifying user credentials that carry this designation will be well known and widely recognized (e.g., Notary Public endorsement).'

To illustrate this convention, consider the following examples. Employees of RSADSI will hold certificates which indicate "RSADSI" as the organization in both the issuer field and the subject field, perhaps with no organizational unit specified. Certificates obtained directly from RSADSI, by user's who are not affiliated with any ON, will also indicate "RSADSI" as the organization and will specify "Notary" as an organizational unit in the issuer field. However, these latter certificates will carry some other designation for organization (and, optionally, organizational unit) in the subject field. Moreover, an organization designated in the subject field for such a certificate will not match any for which RSADSI has an ON registered (to avoid possible confusion).

In all cases described above, when a certificate is generated RSADSI will send a paper reply to the ordering user, including two message hash functions:

1. a message hash computed on the user's identifying information and public component (and sent to RSADSI in the registration process), to guarantee its integrity across the ordering process, and

2. a message hash computed on the public component of RSADSI, to provide independent authentication for this public component which is transmitted to the user via email (see below).

RSADSI will send to the user via electronic mail (not privacy enhanced) a copy of his certificate, a copy of the organization certificate identified in the issuer field of the user's certificate, and the public component used to validate certificates signed by RSADSI. The "issuer" certificate is included to simplify the validation process in the absence of a user-level directory system; its distribution via this procedure will probably be phased out in the future. Thus, as described in RFC-1113, the originator of a message is encouraged, though not required, to include his certificate, and that of its issuer, in the privacy enhanced message header (X-Issuer-Certificate) to ensure that each recipient can process the message using only the information contained in this header. The organization (organizational unit) identified in the subject field of the issuer certificate should correspond to that which the user claims affiliation (as declared in the subject field of his certificate). If there is no appropriate correspondence between these fields, recipients ought to be suspicious of the implied certification path. This relationship should hold except in the case of "non-affiliated" users for whom the "Notary" convention is employed.

In contrast, the issuer field of the issuer's certificate will specify "RSADSI" as the organization, i.e., RSADSI will certify all organizational certificates. This convention allows a recipient to validate any originator's certificate (within the RSADSI certification hierarchy) in just two steps. Even if an organization establishes a certification hierarchy involving organizational units, certificates corresponding to each unit can be certified both by

RSADSI and by the organizational entity immediately superior to the unit in the hierarchy, so as to preserve this short certification path feature. First, the public component of RSADSI is employed to validate the issuer's certificate. Then the issuer's public component is extracted from that certificate and is used to validate the originator's certificate. The recipient then extracts the originator's public component for use in processing the X-Mic-Info field of the message (see and RFC-1113).

The electronic representation used for transmission of the data items described above (between an ON and RSADSI) will be contained in a

subsequent RFC. To verify that the registration process has been successfully completed and to prepare for exchange of privacy-enhanced electronic mail, the user should perform the following steps:

1. extract the RSADSI public component, the issuer's certificate and the user's certificate from the message

2. compute the message hash on the RSADSI public component and compare the result to the corresponding message hash that was included in the paper receipt

3. use the RSADSI public component to validate the signature on the issuer's certificate (RSADSI will be the issuer of this certificate)

4. extract the organization public component from the validated issuer's certificate and use this public component to validate the user certificate

5. extract the identification information and public component from the user's certificate, compute the message hash on it and compare the result to the corresponding message hash value transmitted via the paper receipt

For a user whose order was processed via an ON, successful completion of these steps demonstrates that the certificate issued to him matches that which he requested and which was certified by his ON. It also demonstrates that he possesses the (correct) public component for RSADSI and for the issuer of his certificate. For a user whose order was placed directly with RSADSI, this process demonstrates that his certificate order was properly processed by RSADSI and that he possesses the valid issuer certificate for the RSADSI Notary. The user can use the RSADSI public component to validate organizational certificates for organizations other than his own. He can employ the public component associated with his own organization to validate certificates issued to other users in his organization.

3.3.3.1 Interoperation Across Certification Hierarchy Boundaries

In order to accommodate interoperation with other certification authorities, e.g., foreign or U.S. government CAs, two conventions will be adopted. First, all certifying authorities must agree to "cross-certify" one another, i.e., each must be willing to sign a certificate in which the issuer is that certifying authority and the subject is another certifying authority. Thus, RSADSI might generate a certificate in which it is identified as the issuer and a certifying authority for the U.S. government is indentified as the

subject. Conversely, that U.S. government certifying authority would generate a certificate in which it is the issuer and RSADSI is the subject. This cross-certification of certificates for "top-level" CAs establishes a basis for "lower level" (e.g., organization and user) certificate validation across the hierarchy boundaries. This avoids the need for users in one certification hierarchy to engage in some "out-of-band" procedure to acquire a public-key for use in validating certificates from a different certification hierarchy.

The second convention is that more than one X-Issuer-Certificate field may appear in a privacy-enhanced mail header. Multiple issuer certificates can be included so that a recipient can more easily validate an originator's certificate when originator and recipient are not part of a common CA hierarchy. Thus, for example, if an originator served by the RSADSI certification hierarchy sends a message to a recipient served by a U.S. government hierarchy, the originator could (optionally) include an X-Issuer-Certificate field containing a certificate issued by the U.S. government CA for RSADSI. In this fashion the recipient could employ his public component for the U.S. government CA to validate this certificate for RSADSI, from which he would extract the RSADSI public component to validate the certificate for the originator's organization, from which he would extract the public component required to validate the originator's certificate. Thus, more steps can be required to validate certificates when certification hierarchy boundaries are crossed, but the same basic procedure is employed. Remember that caching of certificates by UAs can significantly reduce the effort required to process messages and so these examples should be viewed as "worse case" scenarios.

3.3.3.2 Certificate Revocation

X.509 states that it is a CA's responsibility to maintain:

1. a time-stamped list of the certificates it issued which have been revoked

2. a time-stamped list of revoked certificates representing other CAs

There are two primary reasons for a CA to revoke a certificate, i.e., suspected compromise of a secret component (invalidating the corresponding public component) or change of user affiliation (invalidating the Distinguished Name). As described in X.509, "hot listing" is one means of propagating information relative to certificate revocation, though it is not a perfect mechanism. In particular, an X.509 Revoked Certificate List (RCL) indicates only the age of the information contained in it; it does not provide any basis for determining if the list is the most current RCL available from a given CA. To help address this concern, the proposed architecture establishes a format for an RCL in which not only the date of issue, but also the next scheduled date of issue is specified. This is a deviation from the format specified in X.509.

Adopting this convention, when the next scheduled issue date arrives a CA must issue a new RCL, even if there are no changes in the list of entries. In this fashion each CA can independently establish and advertise the frequency with which RCLs are issued by that CA. Note that this does not preclude RCL issuance on a more frequent basis, e.g., in case of some emergency, but no Internet-wide mechanisms are architected for alerting users that such an unscheduled issuance has taken place. This scheduled RCL issuance convention allows users (UAs) to determine whether a given RCL is "out of date," a facility not available from the standard RCL format.

A recent (draft) version of the X.509 recommendation calls for each RCL to contain the serial numbers of certificates which have been revoked by the CA administering that list, i.e., the CA that is identified as the issuer for the corresponding revoked certificates. Upon receipt of a RCL, a UA should compare the entries against any cached certificate information, deleting cache entries which match RCL entries. (Recall that the certificate serial numbers are unique only for each issuer, so care must be exercised in effecting this cache search.) The UA should also retain the RCL to screen incoming messages to detect use of revoked certificates carried in these message headers. More specific details for processing RCL are beyond the scope of this RFC as they are a function of local certificate management techniques.

In the architecture defined by this RFC, a RCL will be maintained for each CA (organization or organizational unit), signed using the private component of that organization (and thus verifiable using the public component of that organization as extracted from its certificate). The RSADSI Notary organizational unit is included in this collection of RCLs. CAs operated under the auspices of the U.S. government or foreign CAs are requested to provide RCLs conforming to these conventions, at least until such time as X.509 RCLs provide equivalent functionality, in support of interoperability with the Internet community. An additional, "top level" RCL, will be maintained by RSAD-SI, and should be maintained by other "top level" CAs, for revoked organizational certificates.

The hot listing procedure (expect for this top level RCL) will be effected by having an ON from each organization transmit to RSADSI a list of the serial numbers of users within his organization, to be hot listed. This list will be transmitted using privacy-enhanced

mail to ensure authenticity and integrity and will employ representation conventions to be provided in a subsequent RFC. RSADSI will format the RCL, sign it using the private component of the organization, and transmit it to the ON for dissemination, using a representation defined in a subsequent RFC. Means for dissemination of RCLs, both within the administrative domain of a CA and across domain boundaries, are not specified by this proposal. However, it is anticipated that each hot list will also be available via network information center databases, directory servers, etc.

The following ASN.1 syntax, derived from X.509, defines the format of RCLs for use in the Internet privacy enhanced email environment. See the ASN.1 definition of certificates (later in this RFC or in X.509, Annex G) for comparison.

```
revokedCertificateList  ::=     SIGNED SEQUENCE {
        signature       AlgorithmIdentifier,
        issuer          Name,
        list            SEQUENCE RCLEntry,
        lastUpdate      UTCTime,
        nextUpdate      UTCTime}

RCLEntry        ::=     SEQUENCE {
        subject         CertificateSerialNumber,
        revocationDate  UTCTime}
```

3.4 Certificate Definition and Usage

3.4.1 Contents and Use

A certificate contains the following contents:

1. version
2. serial number
3. certificate signature (and associated algorithm identifier)
4. issuer name
5. validity period
6. subject name
7. subject public component (and associated algorithm identifier)

This section discusses the interpretation and use of each of these certificate elements.

3.4.1.1 Version Number

The version number field is intended to facilitate orderly changes in certificate formats over time. The initial version number for certificates is zero (0).

3.4.1.2 Serial Number

The serial number field provides a short form, unique identifier for each certificate generated by an issuer. The serial number is used in RCLs to identify revoked certificates instead of including entire certificates. Thus each certificate generated by an issuer must contain a unique serial number. It is suggested that these numbers be issued as a compact, monotonic increasing sequence.

3.4.1.3 Subject Name

A certificate provides a representation of its subject's identity and organizational affiliation in the form of a Distinguished Name. The fundamental binding ensured by the privacy enhancement mechanisms is that between public-key and the user identity. CCITT Recommendation X.500 defines the concept of Distinguished Name.

Version 2 of the U.S. Government Open Systems Interconnection Profile (GOSIP) specifies maximum sizes for O/R Name attributes. Since most of these attributes also appear in Distinguished Names, we have adopted the O/R Name attribute size constraints specified in GOSIP and noted below. Using these size constraints yields a maximum Distinguished Name length (exclusive of ASN encoding) of two-hundred fifty-nine (259) characters, based on the required and optional attributes described below for subject names. The following attributes are required in subject Distinguished Names for purposes of this RFC:

1. Country Name in standard encoding (e.g., the two-character Printable String "US" assigned by ISO 3166 as the identifier for the United States of America, the string "GB" assigned as the identifier for the United Kingdom, or the string "NQ" assigned as the identifier for Dronning Maud Land). Maximum ASCII character length of three (3).

2. Organizational Name (e.g., the Printable String "Bolt Beranek and Newman, Inc."). Maximum ASCII character length of sixty-four (64).

3. Personal Name (e.g., the X.402/X.411 structured Printable String encoding for the name John Linn). Maximum ASCII character length of sixty-four (64).

The following attributes are optional in subject Distinguished Names for purposes of this RFC:

1. Organizational Unit Name(s) (e.g., the Printable String "BBN Communications Corporation") A hierarchy of up to four organizational unit names may be provided; the least significant member of the hierarchy is represented first. Each of these attributes has a maximum ASCII character length of thirty-two (32), for a total of one-hundred and twenty-eight (128) characters if all four are present.

3.4.1.4 Issuer Name

A certificate provides a representation of its issuer's identity, in the form of a Distinguished Name. The issuer identification is needed in order to determine the appropriate issuer public component to use in performing certificate validation. The following attributes are required in issuer Distinguished Names for purposes of this RFC:

1. Country Name (e.g., encoding for "US")

2. Organizational Name

The following attributes are optional in issuer Distinguished Names for purposes of this RFC:

1. Organizational Unit Name(s). (A hierarchy of up to four organizational unit names may be provided; the least significant member of the hierarchy is represented first.) If the issuer is vouching for the user identity in the Notary capacity described above, then exactly one instance of this field must be present and it must consist of the string "Notary".

As noted earlier, only organizations are allowed as issuers in the proposed authentication hierarchy. Hence the Distinguished Name for an issuer should always be that of an organization, not a user, and thus no Personal Name field may be included in the Distinguished Name of an issuer.

3.4.1.5 Validity Period

A certificate carries a pair of time specifiers, indicating the start and end of the time period over which a certificate is intended to be used. No message should ever be prepared for transmission with a non-current certificate, but recipients should be prepared to receive messages processed using recently-expired certificates. This fact results from the unpredictable (and sometimes substantial)

transmission delay of the staged-delivery electronic mail environment. The default and maximum validity period for certificates issued in this system will be two years.

3.4.1.6 Subject Public Component

A certificate carries the public component of its associated entity, as well as an indication of the algorithm with which the public component is to be used. For purposes of this RFC, the algorithm identifier will indicate use of the RSA algorithm, as specified in RFC-1115. Note that in this context, a user's public component is actually the modulus employed in RSA algorithm calculations. A "universal" (public) exponent is employed in conjunction with the modulus to complete the system. Two choices of exponents are recommended for use in this context and are described in section 3.4.3. Modulus size will be permitted to vary between 320 and 632 bits.

3.4.1.7 Certificate Signature

A certificate carries a signature algorithm identifier and a signature, applied to the certificate by its issuer. The signature is validated by the user of a certificate, in order to determine that the integrity of its contents have not been compromised subsequent to generation by a CA. An encrypted, one-way hash will be employed as the signature algorithm. Hash functions suitable for use in this context are notoriously difficult to design and tend to be computationally intensive. Initially we have adopted a hash function developed by RSADSI and which exhibits performance roughly equivalent to the DES (in software). This same function has been selected for use in other contexts in this system where a hash function (message hash algorithm) is required, e.g., MIC for multicast messages. In the future we expect other one-way hash functions will be added to the list of algorithms designated for this purpose.

3.4.2 Validation Conventions

Validating a certificate involves verifying that the signature affixed to the certificate is valid, i.e., that the hash value computed on the certificate contents matches the value that results from decrypting the signature field using the public component of the issuer. In order to perform this operation the user must possess the public component of the issuer, either via some integrity-assured channel, or by extracting it from another (validated) certificate. In the proposed architecture this recursive operation is terminated quickly by adopting the convention that RSADSI will certify the certificates of all organizations or organizational units which act as issuers for end users. (Additional validation steps may be

required for certificates issued by other CAs as described in section 3.3.3.1.)

Certification means that RSADSI will sign certificates in which the subject is the organization or organizational unit and for which RSADSI is the issuer, thus implying that RSADSI vouches for the credentials of the subject. This is an appropriate construct since each ON representing an organization or organizational unit must have registered with RSADSI via a procedure more rigorous than individual user registration. This does not preclude an organizational unit from also holding a certificate in which the "parent" organization (or organizational unit) is the issuer. Both certificates are appropriate and permitted in the X.509 framework. However, in order to facilitate the validation process in an environment where user-level directory services are generally not available, we will (at this time) adopt this certification convention.

The public component needed to validate certificates signed by RSADSI (in its role as a CA for issuers) is transmitted to each user as part of the registration process (using electronic mail with independent, postal confirmation via a message hash). Thus a user will be able to validate any user certificate (from the RSADSI hierarchy) in at most two steps. Consider the situation in which a user receives a privacy enhanced message from an originator with whom the recipient has never previously corresponded. Based on the certification convention described above, the recipient can use the RSADSI public component to validate the issuer's certificate contained in the X-Issuer-Certificate field. (We recommend that, initially, the originator include his organization's certificate in this optional field so that the recipient need not access a server or cache for this public component.) Using the issuer's public component (extracted from this certificate), the recipient can validate the originator's certificate contained in the X-Certificate field of the header.

Having performed this certificate validation process, the recipient can extract the originator's public component and use it to decrypt the content of the X-MIC-Info field and thus verify the data origin authenticity and integrity of the message. Of course, implementations of privacy enhanced mail should cache validated public components (acquired from incoming mail or via the message from a user registration process) to speed up this process. If a message arrives from an originator whose public component is held in the recipient's cache, the recipient can immediately employ that public component without the need for the certificate validation process described here. Also note that the arithmetic required for certificate validation is considerably faster than that involved in digitally signing a certificate, so as to minimize the computational burden on users.

A separate issue associated with validation of certificates is a semantic one, i.e., is the entity identified in the issuer field appropriate to vouch for the identifying information in the subject field. This is a topic outside the scope of X.509, but one which must be addressed in any viable system. The hierarchy proposed in this RFC is designed to address this issue. In most cases a user will claim, as part of his identifying information, affiliation with some organization and that organization will have the means and responsibility for verifying this identifying information. In such circumstances one should expect an obvious relationship between the Distinguished Name components in the issuer and subject fields.

For example, if the subject field of a certificate identified an individual as affiliated with the "Widget Systems Division" (Organizational Unit Name) of "Compudigicorp" (Organizational Name), one would expect the issuer field to specify "Compudigicorp" as the Organizational Name and, if an Organizational Unit Name were present, it should be "Widget Systems Division." If the issuer's certificate indicated "Compudigicorp" as the subject (with no Organizational Unit specified), then the issuer should be "RSADSI." If the issuer's certificate indicated "Widget Systems Division" as Organizational Unit and "Compudigicorp" as Organization in the subject field, then the issuer could be either "RSADSI" (due to the direct certification convention described earlier) or "Compudigicorp" (if the organization elected to distribute this intermediate level certificate). In the later case, the certificate path would involve an additional step using the certificate in which "Compudigicorp" is the subject and "RSADSI" is the issuer. One should be suspicious if the validation path does not indicate a subset relationship for the subject and issuer Distinguished Names in the certification path, expect where cross-certification is employed to cross CA boundaries.

It is a local matter whether the message system presents a human user with the certification path used to validate a certificate associated with incoming, privacy-enhanced mail. We note that a visual display of the Distinguished Names involved in that path is one means of providing the user with the necessary information. We recommend, however, that certificate validation software incorporate checks and alert the user whenever the expected certification path relationships are not present. The rationale here is that regular display of certification path data will likely be ignored by users, whereas automated checking with a warning provision is a more effective means of alerting users to possible certification path anomalies. We urge developers to provide facilities of this sort.

3.4.3 Relation with X.509 Certificate Specification

An X.509 certificate can be viewed as two components: contents and an encrypted hash. The encrypted hash is formed and processed as follows:

1. X, the hash, is computed as a function of the certificate contents

2. the hash is signed by raising X to the power e (modulo n)

3. the hash's signature is validated by raising the result of step 2 to the power d (modulo n), yielding X, which is compared with the result computed as a function of certificate contents.

Annex C to X.509 suggests the use of Fermat number F4 (65537 decimal, 1 + 2 **16) as a fixed value for e which allows relatively efficient authentication processing, i.e., at most seventeen (17) multiplications are required to effect exponentiation). As an alternative one can employ three (3) as the value for e, yielding even faster exponentiation, but some precautions must be observed (see RFC-1115). Users of the algorithm select values for d (a secret quantity) and n (a non-secret quantity) given this fixed value for e. As noted earlier, this RFC proposes that either three (3) or F4 be employed as universal encryption exponents, with the choice specified in the algorithm identifier. In particular, use of an exponent value of three (3) for certificate validation is encouraged, to permit rapid certificate validation. Given these conventions, a user's public component, and thus the quantity represented in his certificate, is actually the modulus (n) employed in this computation (and in the computations used to protect the DEK and MSGHASH, as described in RFC-1113). A user's private component is the exponent (d) cited above.

The X.509 certificate format is defined (in X.509, Annex G) by the following ASN.1 syntax:

```
     Certificate ::= SIGNED SEQUENCE{
             version [0]     Version DEFAULT v1988,
             serialNumber    CertificateSerialNumber,
             signature       AlgorithmIdentifier,
             issuer          Name,
             validity        Validity,
             subject         Name,
             subjectPublicKeyInfo    SubjectPublicKeyInfo}

     Version ::=     INTEGER {v1988(0)}

     CertificateSerialNumber ::=     INTEGER
```

```
     Validity ::=    SEQUENCE{
             notBefore       UTCTime,
             notAfter        UTCTime}

     SubjectPublicKeyInfo ::=        SEQUENCE{
             algorithm               AlgorithmIdentifier,
                subjectPublicKey        BIT STRING}

        AlgorithmIdentifier ::= SEQUENCE{
                algorithm       OBJECT IDENTIFIER,
                parameters      ANY DEFINED BY algorithm OPTIONAL}
```

All components of this structure are well defined by ASN.1 syntax defined in the 1988 X.400 and X.500 Series Recommendations, except for the AlgorithmIdentifier. An algorithm identifier for RSA is contained in Annex H of X.509 but is unofficial. RFC-1115 will provide detailed syntax and values for this field.

NOTES:

[1] CCITT Recommendation X.411 (1988), "Message Handling Systems: Message Transfer System: Abstract Service Definition and Procedures".

[2] CCITT Recommendation X.509 (1988), "The Directory Authentication Framework".

Authors' Addresses

Steve Kent
BBN Communications
50 Moulton Street
Cambridge, MA 02138

Phone: (617) 873-3988

EMail: kent@BBN.COM

John Linn
Secure Systems
Digital Equipment Corporation

85 Swanson Road, XB1-2/D04
Boxborough, MA 01719-1326

Phone: 508-264-5491

EMail: Linn@ultra.enet.dec.com